SYNTAX

Matt Post
IntroHLT class
21 October 2019

JOHNS HOPKINS
UNIVERSITY

Fred Jones was worn out from caring for his often screaming and crying wife during the day but he couldn’t sleep at night for fear that she in a stupor from the drugs that didn’t ease the pain would set the house ablaze with a cigarette

- 46 words, 46! permutations of those words, the vast majority of them ungrammatical and meaningless
- How is that we can
  - process and understand this sentence?
  - discriminate it from the sea of ungrammatical permutations it floats in?

Today we will cover

Linguistics	*Computer Science*
what is syntax?	how can a computer find a sentence's structure?
what is a grammar and where do they come from?	what can parse trees be used for?

Today we will cover

Linguistics

Computer Science

what is syntax?

how can a computer find a sentence's structure?

what is a grammar and where do they come from?

what can parse trees be used for?

MORGAN & CLAYPOOL PUBLISHERS

Linguistic Fundamentals for Natural Language Processing

100 Essentials from Morphology and Syntax

Emily M. Bender

SYNTHESIS LECTURES ON HUMAN LANGUAGE TECHNOLOGIES

Graeme Hirst, *Series Editor*

What is syntax?

- A set of constraints on the possible sentences in the language
  - *A set of <u>constraint</u> on the possible <u>sentence.</u>
  - *Dipanjan asked <u>[a]</u> question.
  - *You are <u>on</u> class.

- A finite set of rules licensing an infinite number of strings

What isn't syntax?

- A "scaffolding for meaning" (Weds), but not the same as meaning

grammatical meaningful	grammatical meaningless
ungrammatical meaningful	ungrammatical meaningless

Parts of Speech (POS)

- Three definitions of **noun**

<u>Grammar school</u>
("metaphysical")
a person, place, thing, or idea

Parts of Speech (POS)

- Three definitions of **noun**

Grammar school ("metaphysical")

a person, place, thing, or idea

Distributional

the set of words that have the same distribution as other nouns

{I,you,he} saw the {bird,cat,dog}.

Parts of Speech (POS)

- Three definitions of **noun**

<u>Grammar school</u>
(“metaphysical”)
a person, place, thing, or idea

<u>Distributional</u>
the set of words that have the same distribution as other nouns

{I,you,he} saw the {bird,cat,dog}.

<u>Functional</u>
the set of words that serve as arguments to verbs

verb
noun *adverb*
adjective

POS Examples

- Collapsed form: single POS collects morphological properties (number, gender, case)
  - NN, NNS, NNP, NNPS
  - RB, RBR, RBS, RP
  - VB, VBD, VBG, VBN, VBP, VBZ
- This works fine…in English

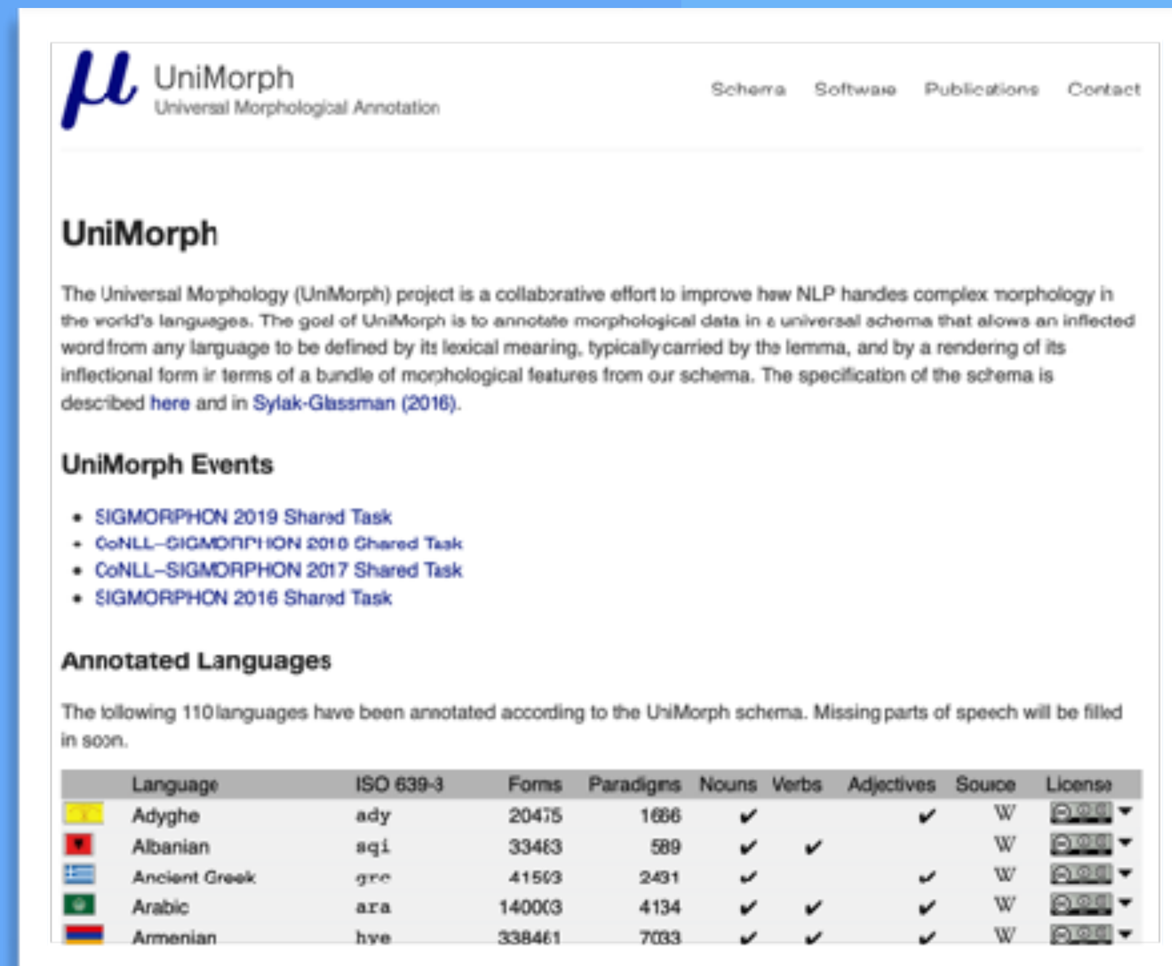
- Collapsing morphological properties doesn’t work so well in other languages
- Attribute-value: morph. properties factored out
  - *Haus:* N[case=nom,number=1,gender=neuter]
  - *Hauses:* N[case=genitive,number=1,gender=neuter]
- In general:
  - Parts of speech are not universal
  - The finer-grained the parts and attributes are, the more language-specific they are
  - Coarse categories will cover more languages

- Two efforts

Unimorph

unimorph.org

unimorph.github.io



UniMorph
Universal Morphological Annotation

Schema Software Publications Contact

UniMorph

The Universal Morphology (UniMorph) project is a collaborative effort to improve how NLP handles complex morphology in the world's languages. The goal of UniMorph is to annotate morphological data in a universal schema that allows an inflected word from any language to be defined by its lexical meaning, typically carried by the lemma, and by a rendering of its inflectional form in terms of a bundle of morphological features from our schema. The specification of the schema is described here and in Sylak-Glassman (2016).

UniMorph Events

- SIGMORPHON 2019 Shared Task
- CoNLL–SIGMORPHON 2018 Shared Task
- CoNLL–SIGMORPHON 2017 Shared Task
- SIGMORPHON 2016 Shared Task

Annotated Languages

The following 110 languages have been annotated according to the UniMorph schema. Missing parts of speech will be filled in soon.

	Language	ISO 639-3	Forms	Paradigms	Nouns	Verbs	Adjectives	Source	License
	Adyghe	ady	20475	1666	✔		✔	W	
	Albanian	sqi	33483	589	✔	✔		W	
	Ancient Greek	grc	41593	2431	✔		✔	W	
	Arabic	ara	140003	4134	✔	✔	✔	W	
	Armenian	hye	338461	7033	✔	✔	✔	W	

A Universal Part-of-Speech Tagset

Slav Petrov[1] Dipanjan Das[2] Ryan McDonald[1]

[1]Google Research, New York, NY, USA, {slav,ryanmcd}@google.com
[2]Carnegie Mellon University, Pittsburgh, PA, USA, dipanjan@cs.cmu.edu

Abstract

To facilitate future research in unsupervised induction of syntactic structure and to standardize best-practices, we propose a tagset that consists of twelve universal part-of-speech categories. In addition to the tagset, we develop a mapping from 25 different treebank tagsets to this universal set. As a result, when combined with the original treebank data, this universal tagset and mapping produce a dataset consisting of common parts-of-speech for 22 different languages. We highlight the use of this resource via three experiments, that (1) compare tagging accuracies across languages, (2) present an unsupervised grammar induction approach that does not use gold standard part-of-speech tags, and (3) use the universal tags to transfer dependency parsers between languages, achieving state-of-the-art results.

Keywords: Part-of-Speech Tagging, Multilinguality, Annotation Guidelines

1. Introduction

Part-of-speech (POS) tagging has received a great deal of attention as it is a critical component of most natural language processing systems. As supervised POS tagging accuracies for English (measured on the PennTreebank (Marcus et al., 1993)) have converged to around 97.3% (Toutanova et al., 2003; Shen et al., 2007; Manning, 2011), the attention has shifted to unsupervised approaches (Christodoulopoulos et al., 2010). In particular, there has been growing interest in both multi-lingual POS induction (Snyder et al., 2009; Naseem et al., 2009) and cross-lingual POS induction via projections (Yarowsky and Ngai, 2001; Xi and Hwa, 2005; Das and Petrov, 2011).

Underlying these studies is the idea that a set of (coarse) syntactic POS categories exists in a similar form across languages. These categories are often called *universals* to represent their cross-lingual nature (Carnie, 2002; Newmeyer, 2005). For example, Naseem et al. (2009) use the Multext-East (Erjavec, 2004) corpus to evaluate their multi-lingual POS induction system, because it uses the same tagset for multiple languages. When corpora with common tagsets are unavailable, a standard approach is to manually define a mapping from language and treebank specific fine-grained tagsets to a predefined universal set. This is the approach taken by Das and Petrov (2011) to evaluate their cross-lingual POS projection system.

To facilitate future research and to standardize best-practices, we propose a tagset that consists of twelve universal POS categories. While there might be some controversy about what the exact tagset should be, we feel that these twelve categories cover the most frequent part-of-speech that exist in most languages. In addition to the tagset, we also develop a mapping from fine-grained POS tags for 25 different treebanks to this universal set. As a result, when combined with the original treebank data, this universal tagset and mapping produce a dataset consisting of common parts-of-speech for 22 different languages.[1] Both the tagset and mappings are made available for download at **http://code.google.com/p/universal-pos-tags/**.

This resource serves multiple purposes. First, as mentioned previously, it is useful for building and evaluating unsupervised and cross-lingual taggers and parsers. Second, it permits for a better comparison of accuracy across languages for supervised taggers. Statements of the form "POS tagging for language X is harder than for language Y" are vacuous when the tagsets used for the two languages are incomparable (not to mention of different cardinality). Finally, it also permits language technology practitioners to train POS taggers with common tagsets across multiple languages. This in turn facilitates downstream application development as there is no need to maintain language specific rules or systems due to differences in treebank annotation guidelines.

In this paper, we specifically highlight three use cases of this resource. First, using our universal tagset and mapping, we run an experiment comparing POS tagging accuracies for 25 different treebanks on a single tagset. Second, we combine the cross-lingual projection part-of-speech taggers of Das and Petrov (2011) with the grammar induction system of Naseem et al. (2010) – which requires a universal tagset – to produce a completely unsupervised grammar induction system for multiple languages, that does not require gold POS tags or any other type of manual annotation in the target language. Finally, we show that a delexicalized English parser, whose predictions rely solely on the universal POS tags of the input sentence, can be used to parse a foreign language POS sequence, achieving higher accuracies than state-of-the-art unsupervised parsers. These experiments highlight that our universal tagset captures a substantial amount of information and carries that information over across languages boundaries.

2. Tagset

While there might be some disagreement about the exact definition of an universal POS tagset (Evans and Levinson, 2009), several scholars have argued that a set of coarse POS categories (or syntactic universals) exists across languages in one form or another (Carnie, 2002; Newmeyer, 2005). Rather than attempting to define an 'a priori' or 'inherent'

[1]We include mappings for two different Chinese, German and Japanese treebanks.

2089

A Universal Part-of-Speech Tagset

Petrov et al. (LREC 2012)

http://www.lrec-conf.org/proceedings/lrec2012/pdf/274_Paper.pdf

Phrases and Constituents

- Longer sequences of words can perform the same function as individual parts of speech:
  - I saw [a kid]
  - I saw [a kid playing basketball]
  - I saw [a kid playing basketball alone on the court]
- This gives rise to the idea of a *phrasal constituent*, which function as a unit in relation to the rest of the sentence

- Tests (Bender #51)
  - *coordination*
    - Kim [read a book], [gave it to Sandy], and [left].
  - *substitution with a word*
    - Kim read [a very interesting book about grammar].
    - Kim read [it].

Heads, arguments, & adjuncts

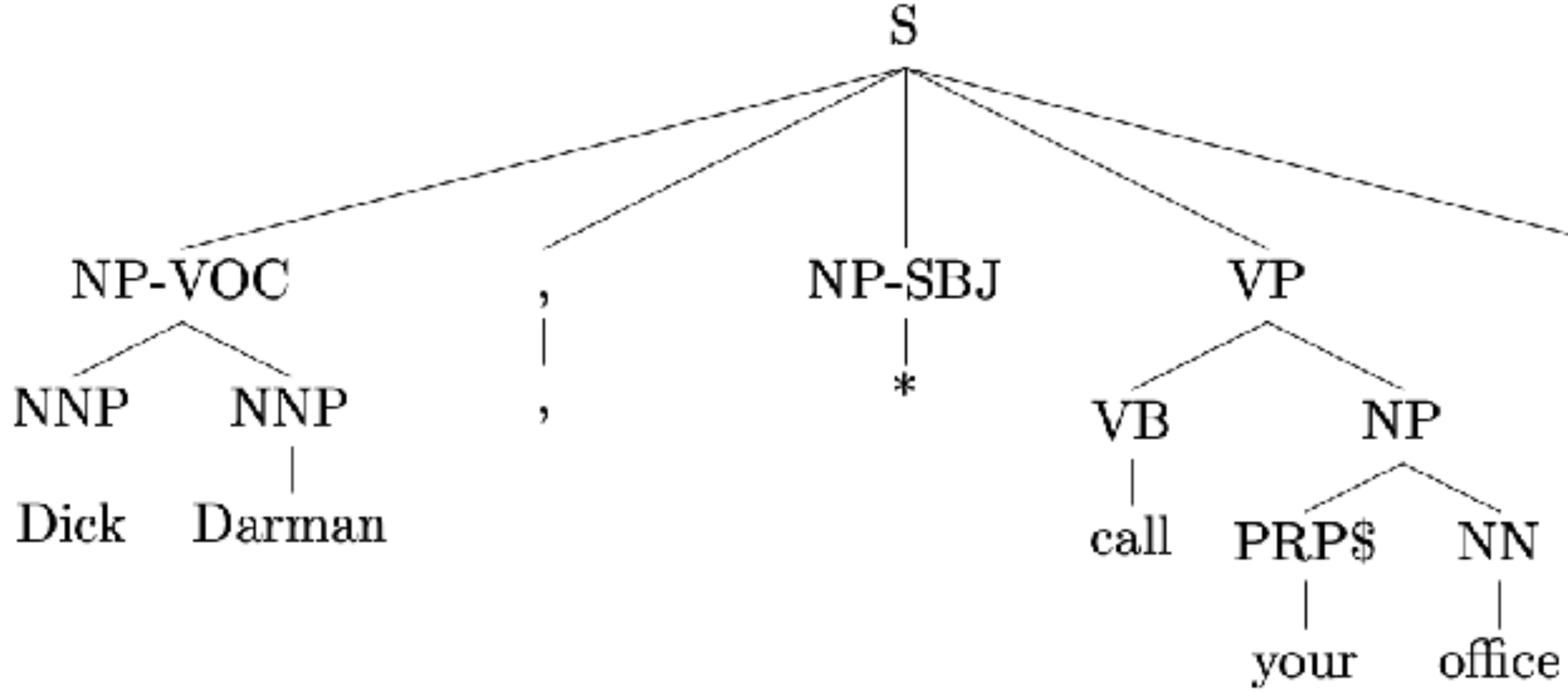
- *Head*: “the sub-constituent which determines the internal structure and external distribution of the constituent as a whole” (Bender #52)
  - Kim planned [to **give** Sandy books].
  - *Kim planned [to **give** Sandy].
  - Kim planned [to give books].
  - *Kim planned [to **see** Sandy books].
  - Kim [**would** [**give** Sandy books]].
  - Pat [**helped** [Kim **give** Sandy books]].
  - *[[**Give** Sandy books] [**surprised** Kim]].

- *Dependents* of a head*:*
  - *Arguments:* selected/licensed by the head and **complete** the meaning
  - *Adjuncts:* not selected and **refine** the meaning
- Examples
  - ADJ
    - Kim is [**ready**$_{\text{ADJ}}$ [to make a pizza]$_{\text{V}}$].
    - *Kim is [**tired**$_{\text{ADJ}}$ [to make a pizza]$_{\text{V}}$].
  - N
    - [The [red]$_{\text{ADJ}}$ **ball**]
    - *[The [red]$_{\text{ADJ}}$ ball [the stick]$_{\text{N}}$]

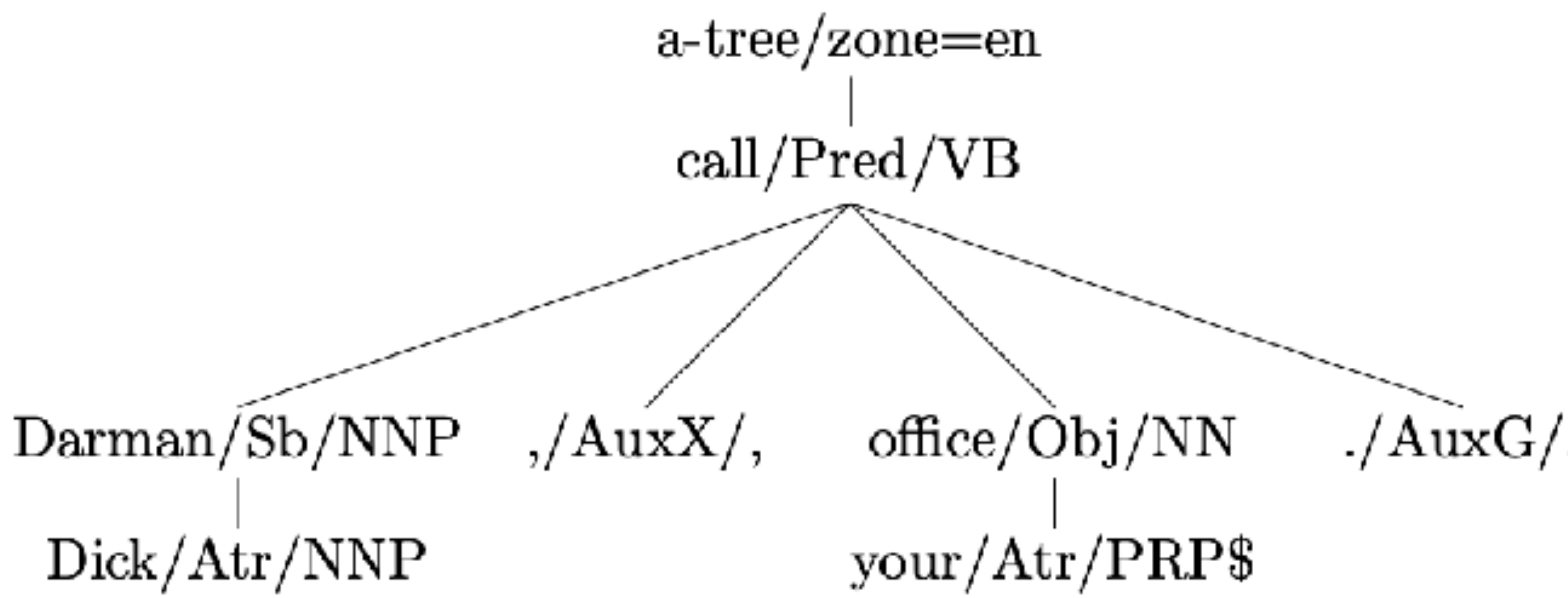
Formalisms

- **Phrase-structure** and **dependency** grammars
  - Phrase-structure: encodes the phrasal components of language
  - Dependency grammars encode the relationships between words

- Phrase / constituent structure
  "*Dick Darman, call your office.*"



- Dependency structure
  "*Dick Darman, call your office.*"



Summary

what is syntax?

A finite set of rules licensing an infinite number of strings

We don't know the rules, but we know that they exist, and native speaker judgments can be used to empirically explore them

Today we will cover

Linguistics

what is syntax?

what is a grammar and where do they come from?

Computer Science

how can a computer find a sentence's structure?

what can parse trees be used for?

Treebanks

- Collections of natural text that are annotated according to a particular syntactic theory
  - Ideally as large as possible
  - Usually annotated by linguistic experts
  - Theories are usually coarsely divided into *constituent* or *dependency* structure

Dependency Treebanks

https://universaldependencies.org

Universal Dependencies

- Dependency trees annotated across languages in a consistent manner

Penn Treebank (1993)

https://catalog.ldc.upenn

- ABOUT
- MEMBERS
- COMMUNICATIONS
- LANGUAGE RESOURCES
- Data
  - Obtaining Data
  - Catalog
  - By Year
  - Top Ten Corpora
  - Projects
  - Search
  - Memberships
  - Data Scholarships
- Tools
- Papers
- LR Wiki
- DATA MANAGEMENT
- COLLABORATIONS

Home › Language Resources › Data

Treebank-3

Item Name:	Treebank-3
Author(s):	Mitchell P. Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz, Ann Taylor
LDC Catalog No.:	LDC99T42
ISBN:	1-58563-163-9
ISLRN:	141-282-691-413-2
Member Year(s):	1999
DCMI Type(s):	Text
Data Source(s):	telephone speech, newswire, microphone speech, transcribed speech, varied
Project(s):	TIDES, GALE
Application(s):	parsing, natural language processing, tagging
Language(s):	English
Language ID(s):	eng
License(s):	LDC User Agreement for Non-Members
Online Documentation:	LDC99T42 Documents
Licensing Instructions:	Subscription & Standard Members, and Non-Members
Citation:	Marcus, Mitchell, et al. Treebank-3 LDC99T42. Web Download. Philadelphia: Linguistic Data Consortium, 1999.
Related Works:	View

Introduction

This release contains the following Treebank-2 Material:

- One million words of 1989 Wall Street Journal material annotated in Treebank II style.
- A small sample of ATIS-3 material annotated in Treebank II style.
- A fully tagged version of the Brown Corpus.

and the following new material:

- Switchboard tagged, dysfluency-annotated, and parsed text
- Brown parsed text

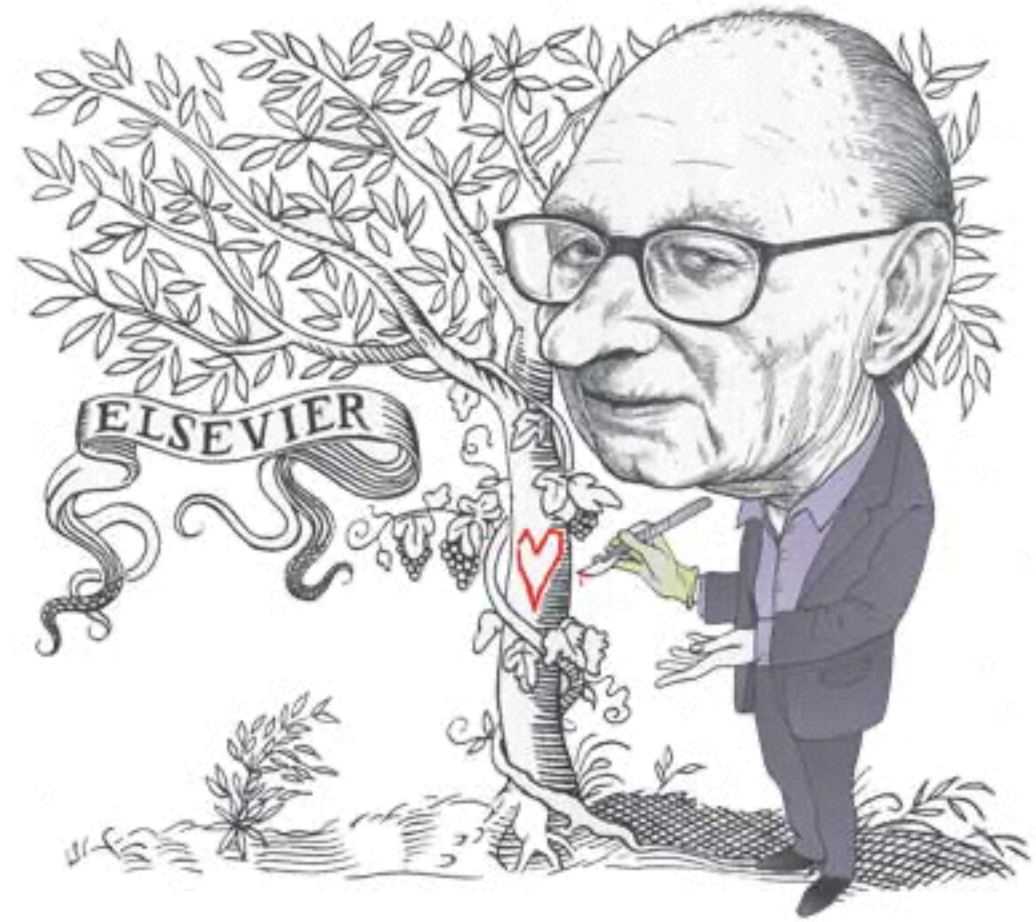
The Treebank bracketing style is designed to allow the extraction of simple predicate/argument structure. Over one million words of text are provided with this bracketing applied.

Data

The Penn Treebank

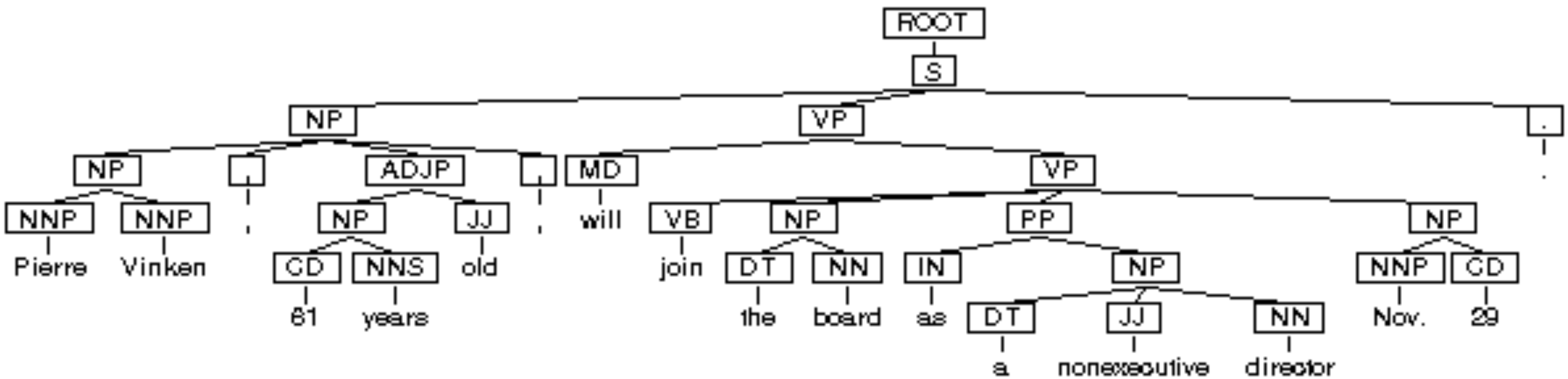
- Syntactic annotation of a million words of the 1989 Wall Street Journal plus other corpora
  - People often discuss “The Penn Treebank” when the mean the WSJ portion of it
- Contains 74 total tags: 36 parts of speech, 7 punctuation tags, and 31 phrasal constituent tags, plus some relation markings
- Was the foundation for an entire field of research and applications for over twenty years

```
( (S
    (NP-SBJ
      (NP (NNP Pierre) (NNP Vinken) )
      (, ,)
      (ADJP
        (NP (CD 61) (NNS years) )
        (JJ old) )
      (, ,) )
    (VP (MD will)
      (VP (VB join)
        (NP (DT the) (NN board) )
        (PP-CLR (IN as)
          (NP (DT a) (JJ nonexecutive) (NN director) ))
        (NP-TMP (NNP Nov.) (CD 29) )))
    (. .) ))
```



https://commons.wikimedia.org/wiki/File:PierreVinken.jpg

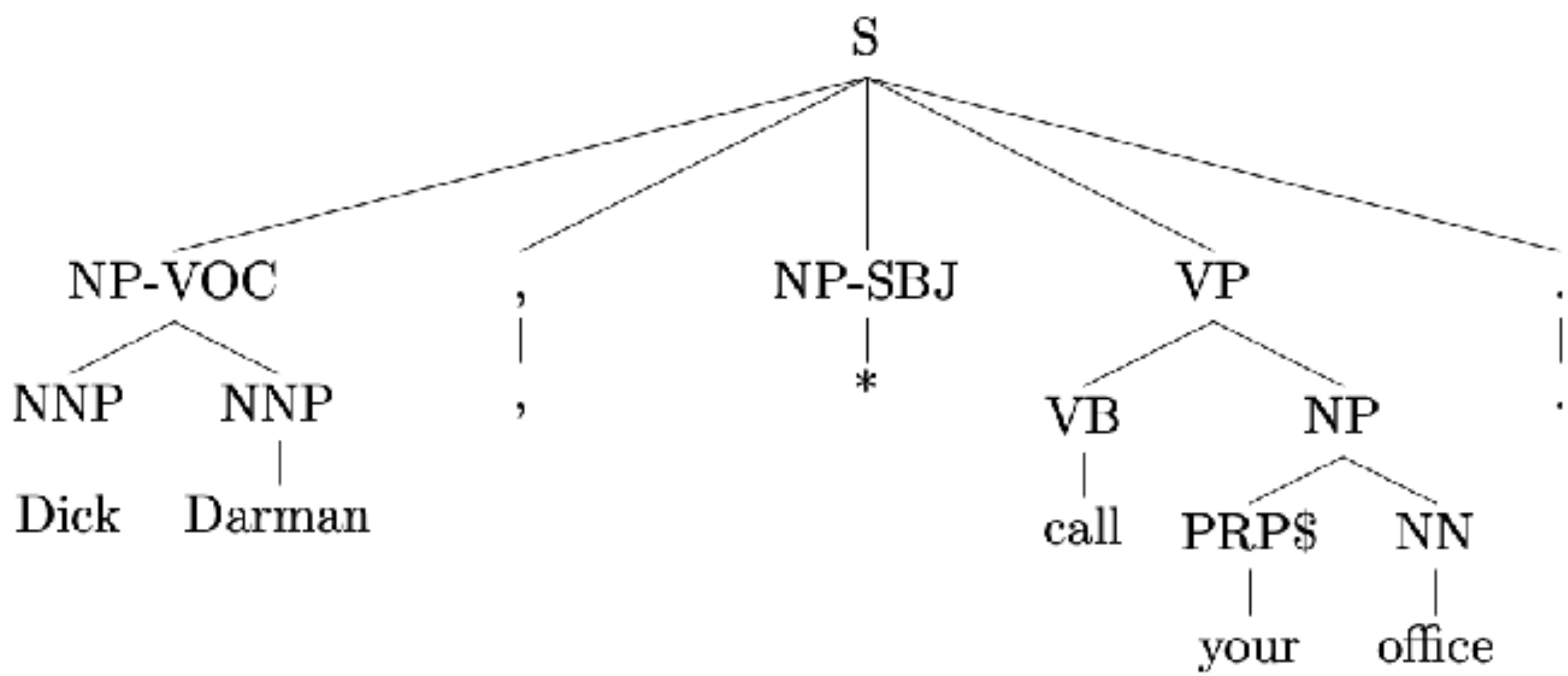
Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29.


ROOT
S
NP
VP
.
NP
,
ADJP
,
MD
VP
NNP
NNP
NP
JJ
will
VB
NP
PP
NP
Pierre
Vinken
CD
NNS
old
join
DT
NN
IN
NP
NNP
CD
61
years
the
board
as
DT
JJ
NN
Nov.
29
a
nonexecutive
director

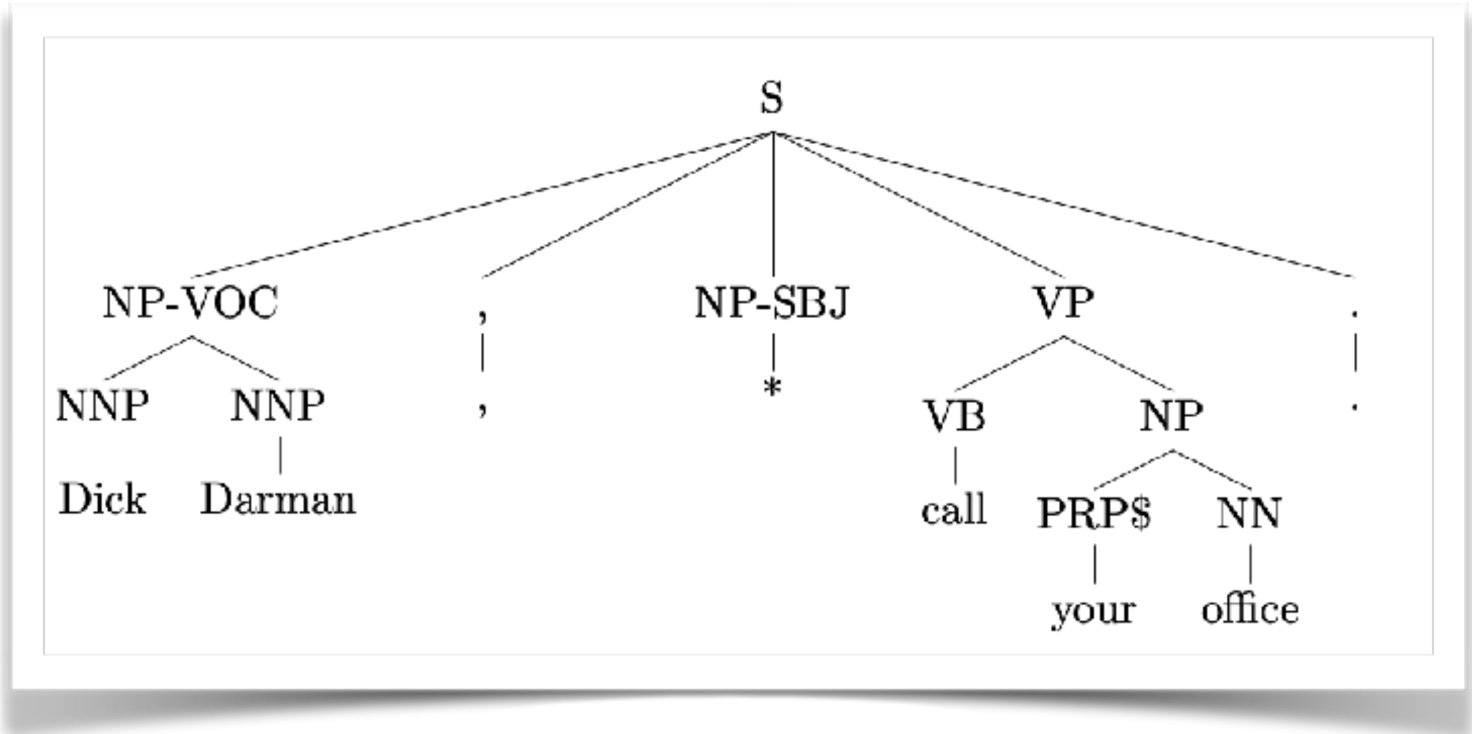
x 49,208

Grammar (one definition)

- A productive mechanism that tells a story about how a Treebank was produced
- How was this tree produced?



- One story:
  - S → NP , NP VP .
  - NP → NNP NNP
  - , → ,
  - NP → *
  - VP → VB NP
  - NP → PRP$ NN
  - . → .
- This is a *top-down, generative* story



Context Free Grammar

- Nonterminals are rewritten based on the lefthand side alone

Chomsky formal language hierarchy

Turing machine
context-sensitive grammar
context free grammar
finite state machine

Context Free Grammar

- Nonterminals are rewritten based on the lefthand side alone
- Algorithm:

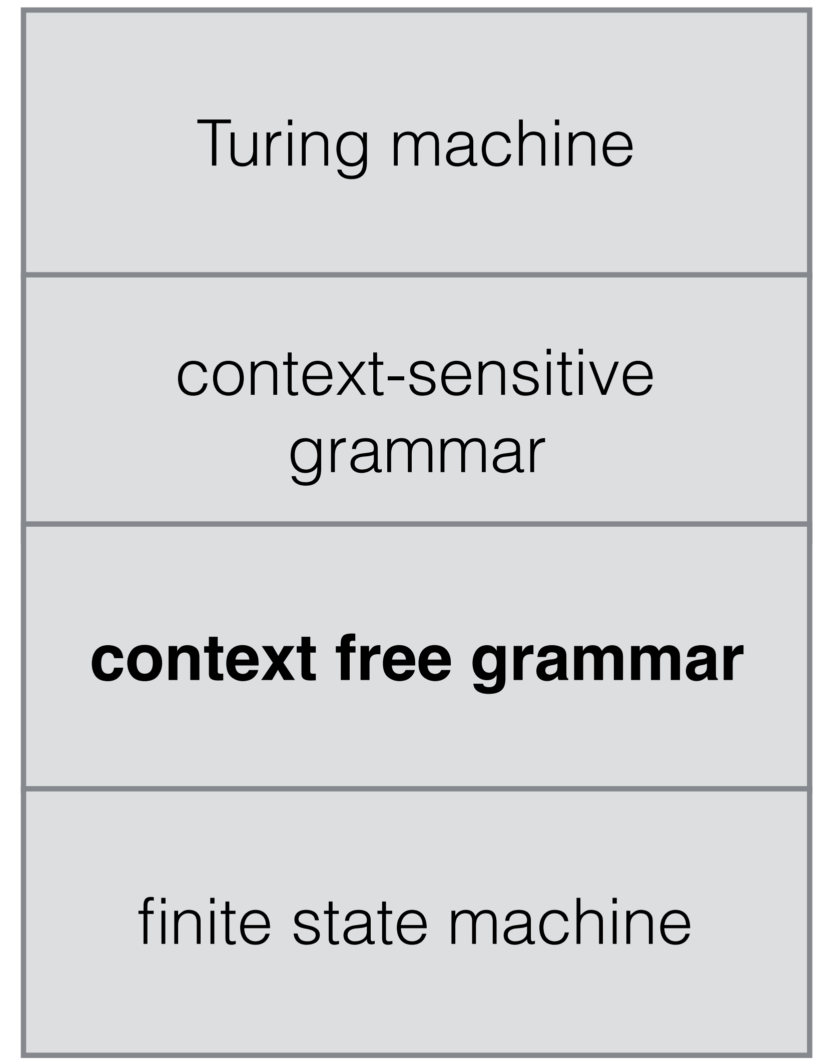
Chomsky formal language hierarchy

Turing machine
context-sensitive grammar
context free grammar
finite state machine

Context Free Grammar

- Nonterminals are rewritten based on the lefthand side alone
- Algorithm:
  - Start with TOP

Chomsky formal language hierarchy



Context Free Grammar

- Nonterminals are rewritten based on the lefthand side alone
- Algorithm:
  - Start with TOP
  - For each leaf nonterminal:

Chomsky formal language hierarchy

Turing machine

context-sensitive grammar

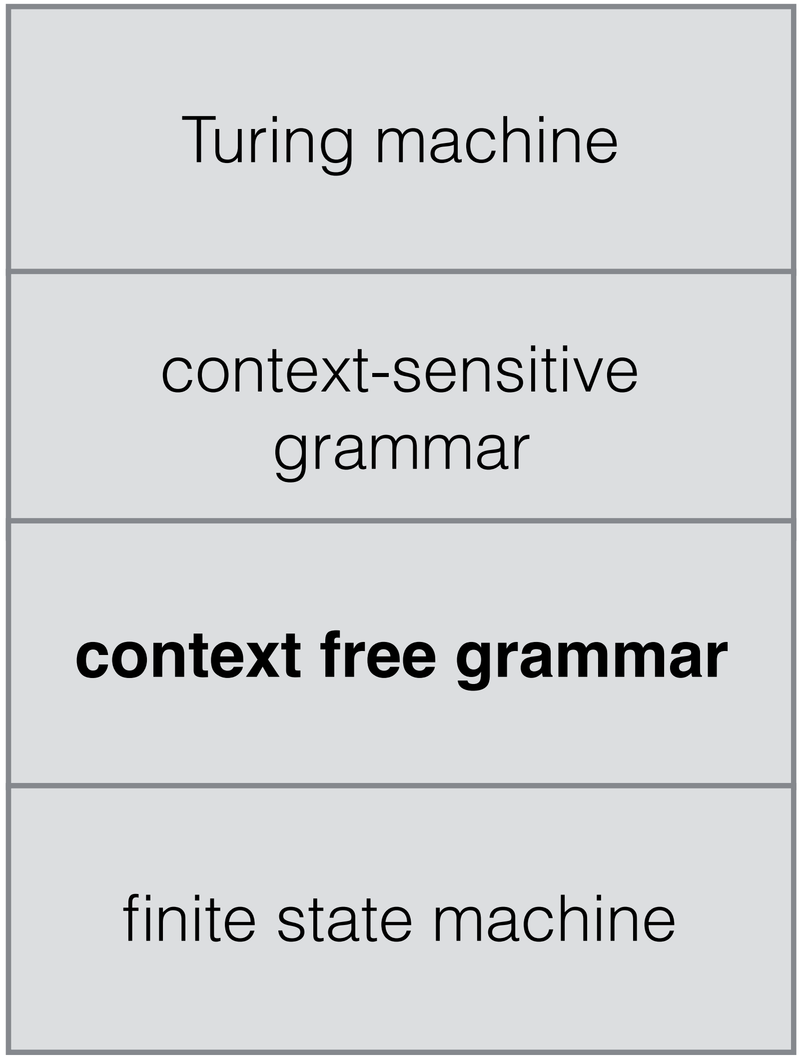
context free grammar

finite state machine

Context Free Grammar

- Nonterminals are rewritten based on the lefthand side alone
- Algorithm:
  - Start with TOP
  - For each leaf nonterminal:
    - Sample a rule from the set of rules for that nonterminal

Chomsky formal language hierarchy



Context Free Grammar

- Nonterminals are rewritten based on the lefthand side alone
- Algorithm:
  - Start with TOP
  - For each leaf nonterminal:
    - Sample a rule from the set of rules for that nonterminal
    - Replace it with

Chomsky formal language hierarchy

Turing machine

context-sensitive grammar

context free grammar

finite state machine

Context Free Grammar

- Nonterminals are rewritten based on the lefthand side alone
- Algorithm:
  - Start with TOP
  - For each leaf nonterminal:
    - Sample a rule from the set of rules for that nonterminal
    - Replace it with
    - Recurse

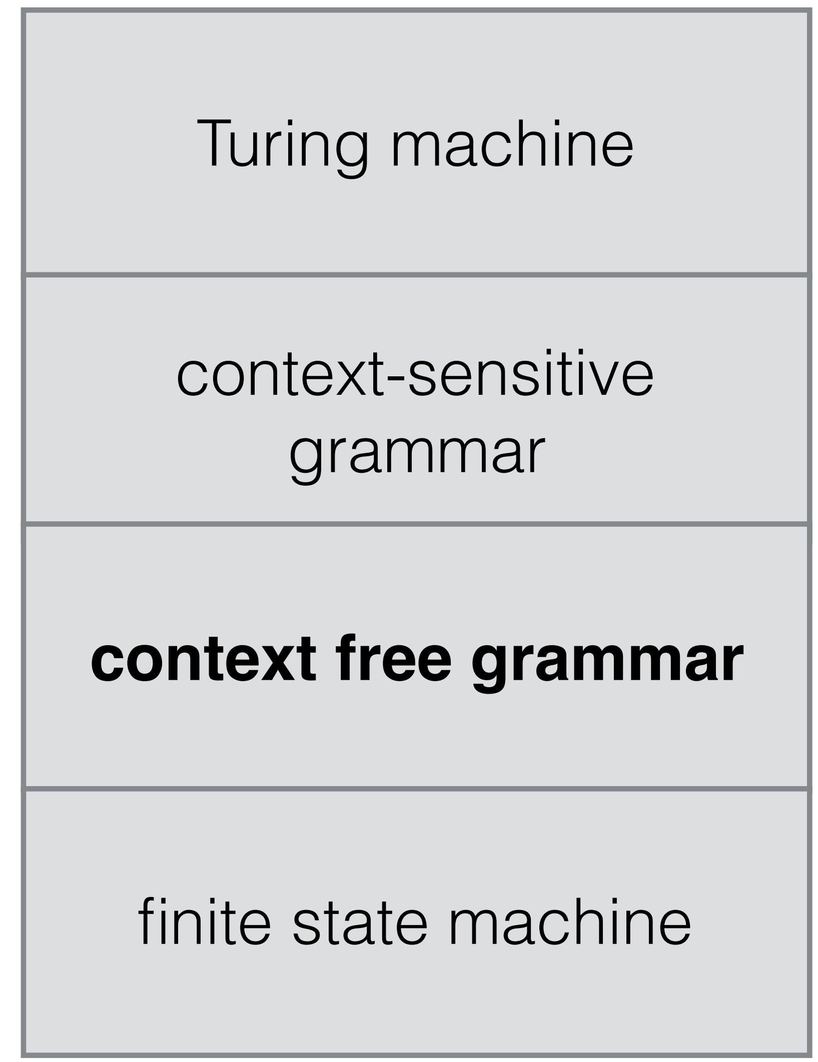
Chomsky formal language hierarchy

Turing machine
context-sensitive grammar
context free grammar
finite state machine

Context Free Grammar

- Nonterminals are rewritten based on the lefthand side alone
- Algorithm:
  - Start with TOP
  - For each leaf nonterminal:
    - Sample a rule from the set of rules for that nonterminal
    - Replace it with
    - Recurse
- Terminates when there are no more nonterminals

Chomsky formal language hierarchy



TOP

TOP → S

TOP

S

TOP → S

S → VP

TOP

S

VP

TOP → S

S → VP

VP → (VB→halt) NP PP

TOP

S

VP

halt NP PP

TOP → S

S → VP

VP → (VB→halt) NP PP

NP → (DT The)
(JJ→market-jarring)
(CD→25)

TOP

S

VP

halt NP PP

halt **The market-jarring 25** PP

TOP → S

S → VP

VP → (VB→halt) NP PP

NP → (DT The)
(JJ→market-jarring)
(CD→25)

PP → (IN→at) NP

TOP

S

VP

halt NP PP

halt **The market-jarring 25** PP

halt The market-jarring 25 **at NP**
(NN→bond)

TOP → S

S → VP

VP → (VB→halt) NP PP

NP → (DT The)
(JJ→market-jarring)
(CD→25)

PP → (IN→at) NP

NP → (DT→the)

TOP	TOP → S
S	S → VP
VP	VP → (VB→halt) NP PP
halt NP PP	NP → (DT The) (JJ→market-jarring) (CD→25)
halt **The market-jarring 25** PP	PP → (IN→at) NP
halt The market-jarring 25 **at NP** (NN→bond)	NP → (DT→the)
halt The market-jarring 25 at **the bond**	

TOP

S

VP

halt NP PP

halt **The market-jarring 25** PP

halt The market-jarring 25 **at NP** (NN→bond)

halt The market-jarring 25 at **the bond**

TOP → S

S → VP

VP → (VB→halt) NP PP

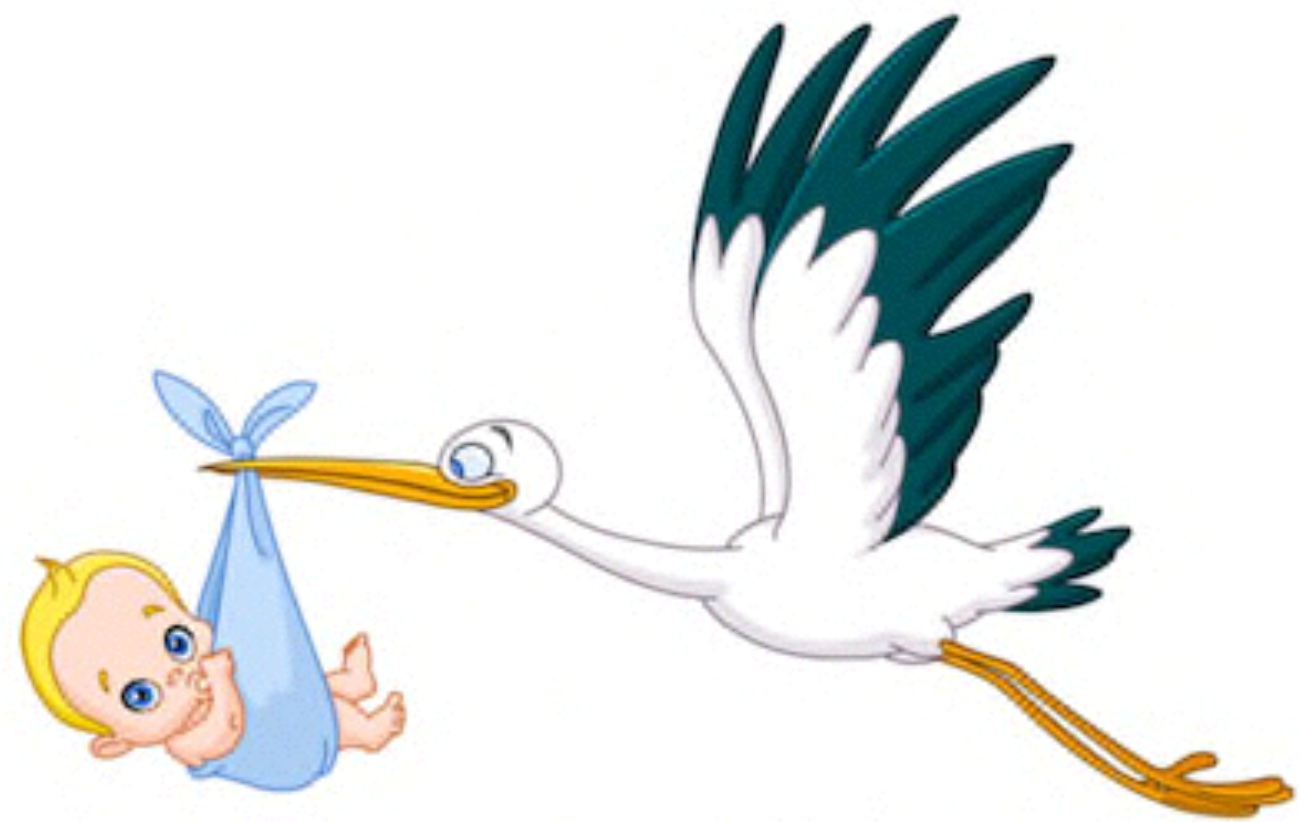
NP → (DT The) (JJ→market-jarring) (CD→25)

PP → (IN→at) NP

NP → (DT→the)

(TOP (S (VP (VB halt) (NP (DT The) (JJ market-jarring) (CD 25)) (PP (IN at) (NP (DT the) (NN bond))))))

Where do grammars come from?



https://www.shutterstock.com/image-vector/stork-carrying-baby-boy-133823486

- The Treebank!
  - Depending on the formalism, it can be read from annotated treebanks
  - Might require additional information
    - e.g., head rules for a dependency grammar conversion
  - This defines a model of how the Treebank was produced

Probabilities

- A useful addition:
  - S → NP , NP VP . [0.002]
  - NP → NNP NNP [0.037]
  - , → , [0.999]
  - NP → * [X]
  - VP → VB NP [0.057]
  - NP → PRP$ NN [0.008]
  - . → . [0.987]
- This is a ***probabilistic**, top-down, generative* story
- Can also be taken from Treebanks $$P(X) = \sum_{X' \in N} \frac{P(X)}{P(X')}$$

Other tasks and models

- *Grammar induction:* humans learn grammar without a Treebank; can computers?
- *Lexicalized models:* build richer models that account for head-driven structure generation
- *Dependency conversions:* define generative dependency process with labeled arcs
- *More descriptive grammars:* (mildly) context-sensitive grammars, attribute-value structures
- *And many more*

Today we will cover

A grammar is an explicit set of rules that explain how a Treebank might have been generated

Grammars come from linguists, either indirectly (via a formalism applied to a Treebank) or directly

what is a grammar and where do they come from?

Today we will cover

Linguistics

what is syntax?

what is a grammar and where do they come from?

Computer Science

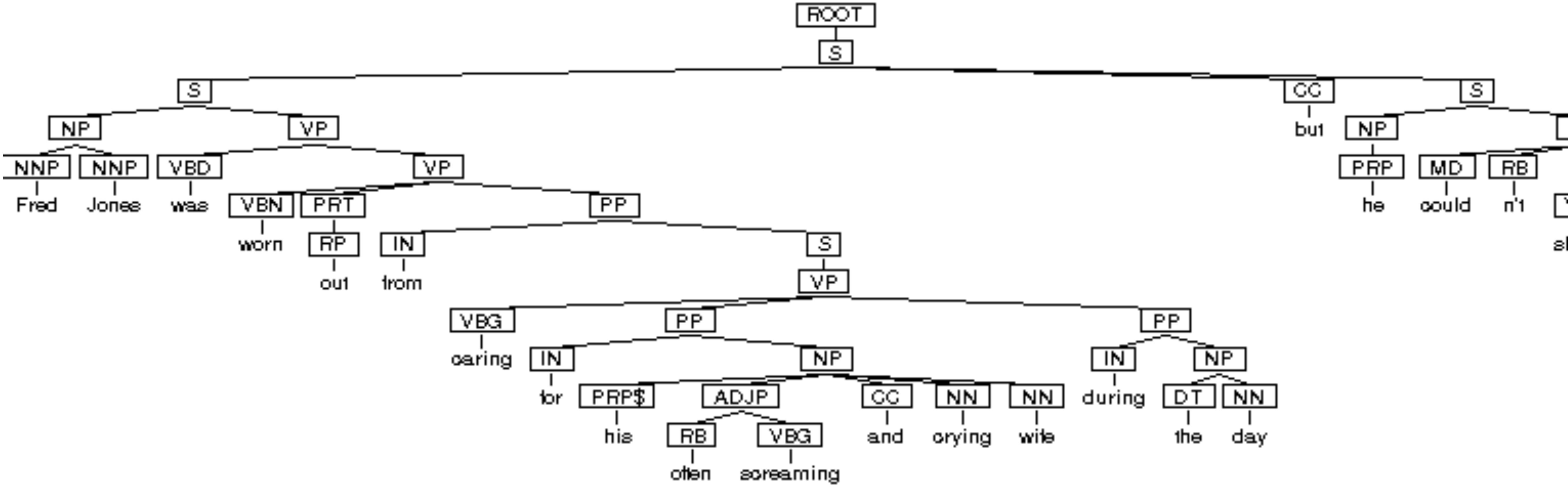
how can a computer find a sentence's structure?

what can parse trees be used for?

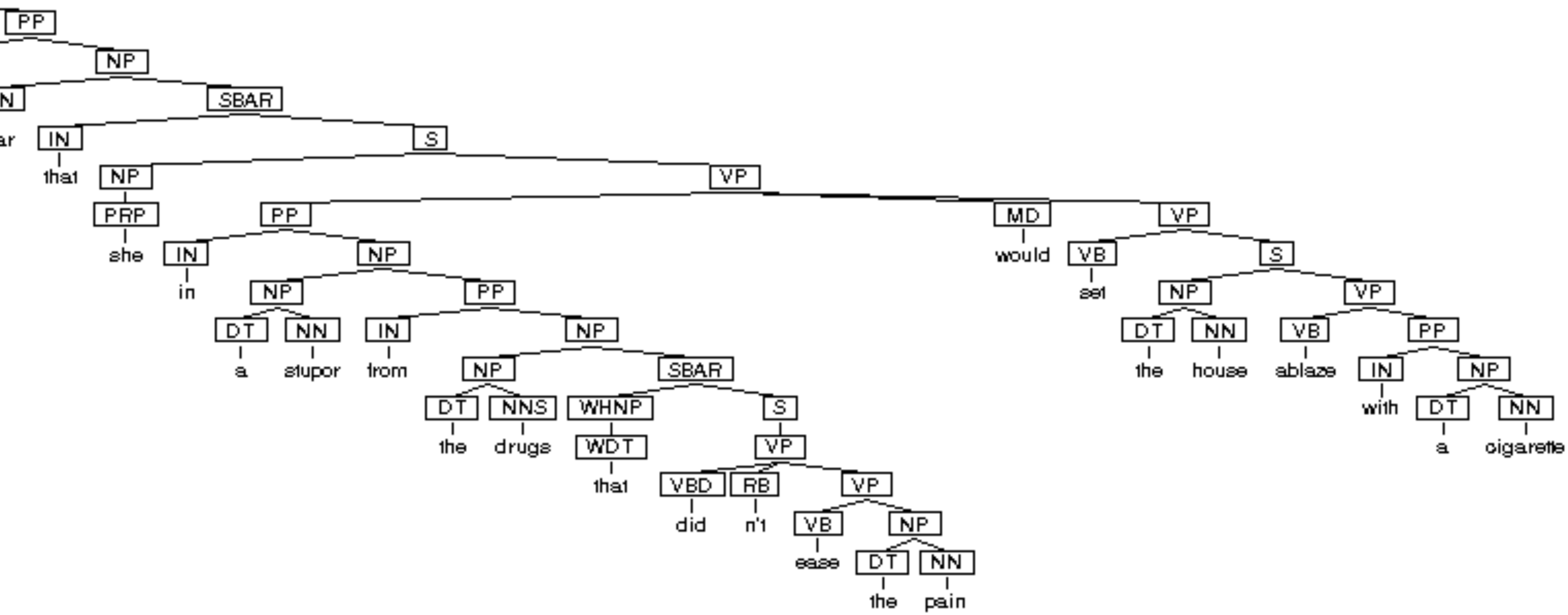
Parsing

- How do we transform

  Fred Jones was worn out from caring for his often screaming and crying wife during the day but he couldn't sleep at night for fear that she in a stupor from the drugs that didn't ease the pain would set the house ablaze with a cigarette.

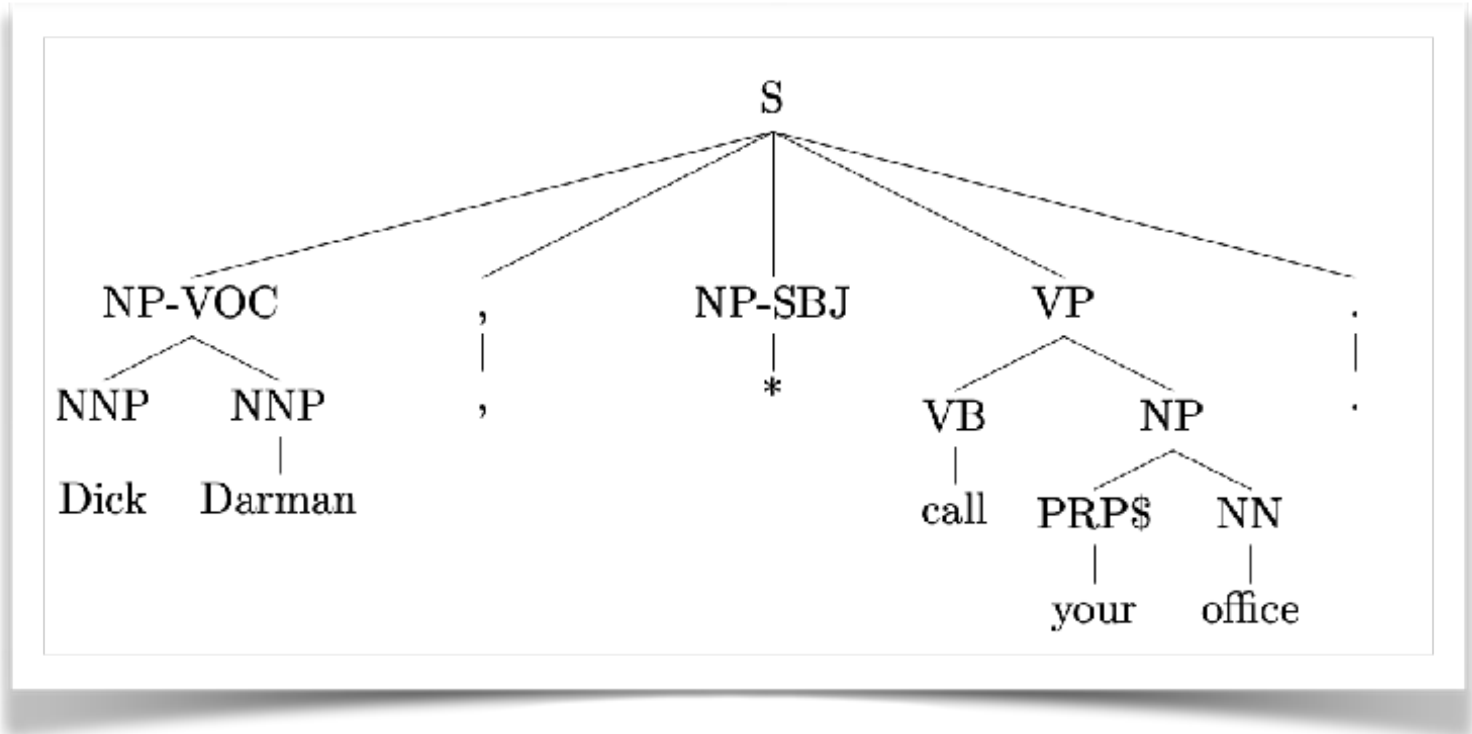


ROOT
S
S
CC
S
NP
VP
but
NP
NNP
NNP
VBD
VP
PRP
MD
RB
Fred
Jones
was
VBN
PRT
PP
he
could
n't
worn
RP
IN
S
out
from
VP
VBG
PP
PP
caring
IN
NP
IN
NP
for
PRP$
ADJP
CC
NN
NN
during
DT
NN
his
RB
VBG
and
crying
wife
the
day
often
screaming



PP
NP
SBAR
IN
that
S
NP
PRP
she
VP
PP
IN
in
NP
NP
DT
a
NN
stupor
PP
IN
from
NP
NP
DT
the
NNS
drugs
SBAR
WHNP
WDT
that
S
VP
VBD
did
RB
n't
VP
VB
ease
NP
DT
the
NN
pain
MD
would
VP
VB
set
S
NP
DT
the
NN
house
VP
VB
ablaze
PP
IN
with
NP
DT
a
NN
cigarette

- One story:
  - S → NP , NP VP .
  - NP → NNP NNP
  - , → ,
  - NP → *
  - VP → VB NP
  - NP → PRP$ NN
  - . → .



- This is a *top-down, generative* story

Chart Parsing

- Cocke-Younger-Kasami (CYK / CKY algorithm)

Time flies like an arrow

0 1 2 3 4 5

Chart Parsing

- Cocke-Younger-Kasami (CYK / CKY algorithm)

NN	*NN,VB*	*VB,IN*	*DT*	*NN*
Time	*flies*	*like*	*an*	*arrow*

0 *1* *2* *3* *4* *5*

Chart Parsing

- Cocke-Younger-Kasami (CYK / CKY algorithm)

$NP \rightarrow NN$ $NP \rightarrow DT\ NN$

NN *NN,VB* *VB,IN* *DT* *NN*

Time *flies* *like* *an* *arrow*

0 *1* *2* *3* *4* *5*

Chart Parsing

- Cocke-Younger-Kasami (CYK / CKY algorithm)

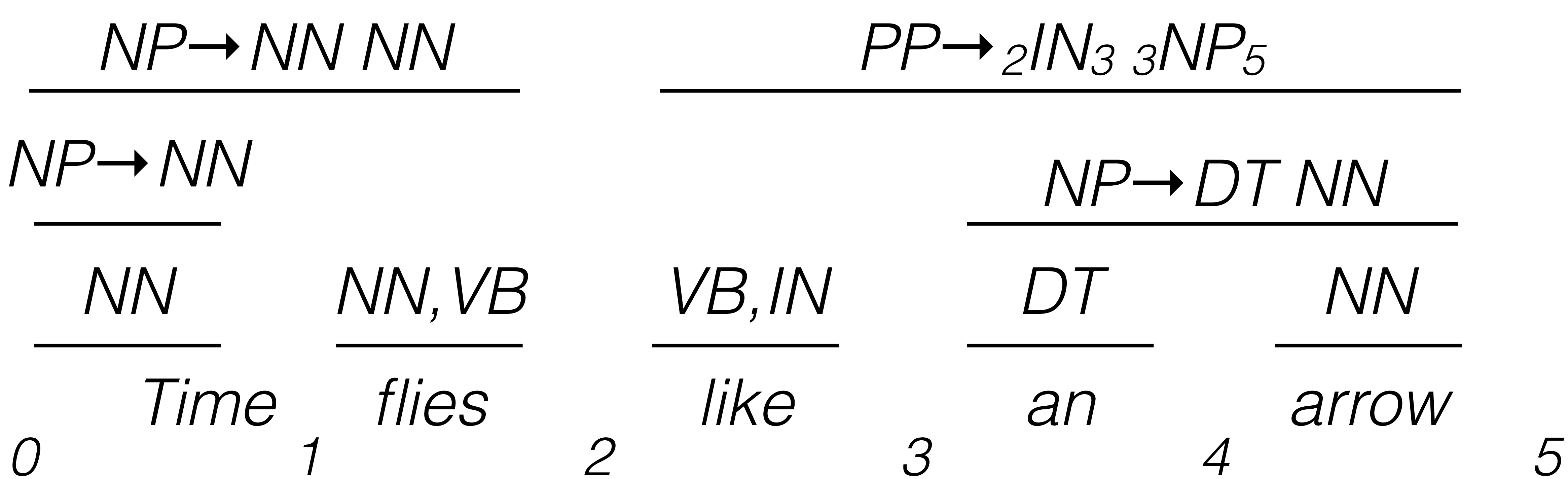


Chart Parsing

- Cocke-Younger-Kasami (CYK / CKY algorithm)

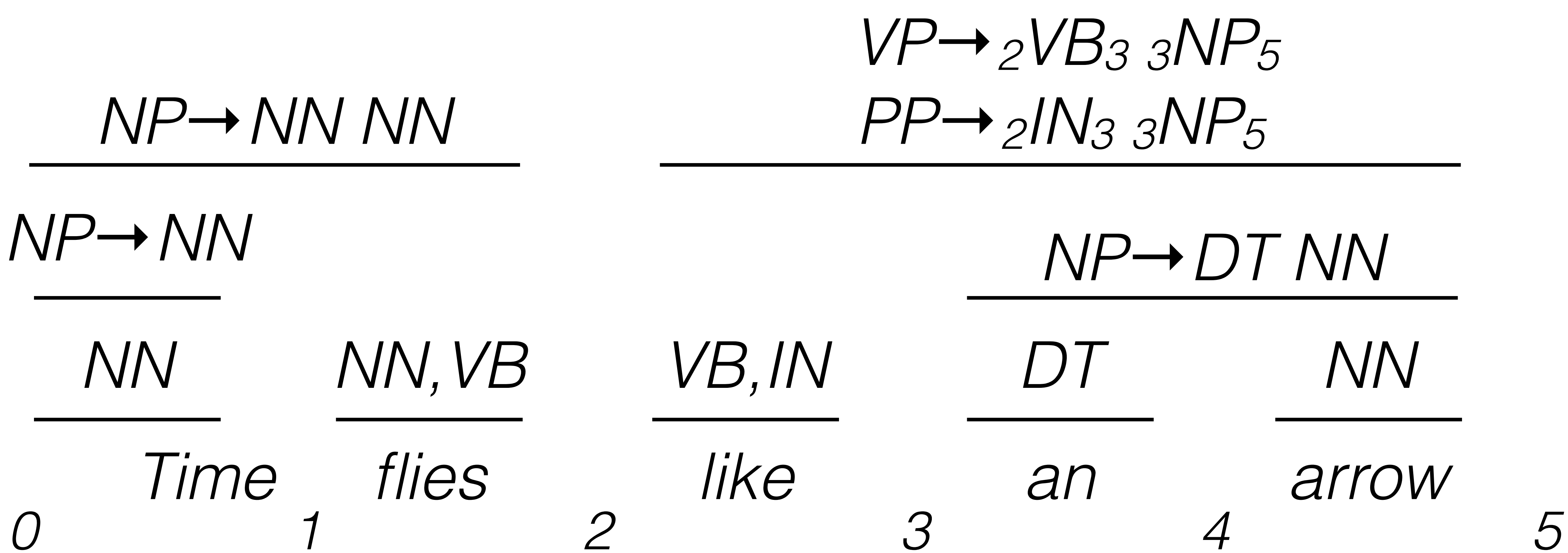


Chart Parsing

- Cocke-Younger-Kasami (CYK / CKY algorithm)

$VP \rightarrow VB\ PP$

$VP \rightarrow {}_2VB_3\ {}_3NP_5$

$PP \rightarrow {}_2IN_3\ {}_3NP_5$

$NP \rightarrow NN\ NN$

$NP \rightarrow NN$

$NP \rightarrow DT\ NN$

NN	*NN,VB*	*VB,IN*	*DT*	*NN*
Time	*flies*	*like*	*an*	*arrow*

0 1 2 3 4 5

Chart Parsing

- Cocke-Younger-Kasami (CYK / CKY algorithm)

$S \rightarrow {}_0NP_1\ {}_1VP_5$

$VP \rightarrow VB\ PP$

$VP \rightarrow {}_2VB_3\ {}_3NP_5$

$PP \rightarrow {}_2IN_3\ {}_3NP_5$

$NP \rightarrow NN\ NN$

$NP \rightarrow NN$

$NP \rightarrow DT\ NN$

NN NN, VB VB, IN DT NN

Time   *flies*   *like*   *an*   *arrow*

0 1 2 3 4 5

Chart Parsing

- Cocke-Younger-Kasami (CYK / CKY algorithm)

$S \rightarrow {}_0NP_2\ {}_2VP_5$

$S \rightarrow {}_0NP_1\ {}_1VP_5$

$VP \rightarrow VB\ PP$

$VP \rightarrow {}_2VB_3\ {}_3NP_5$

$NP \rightarrow NN\ NN$ $\quad$ $PP \rightarrow {}_2IN_3\ {}_3NP_5$

$NP \rightarrow NN$

$NP \rightarrow DT\ NN$

NN $\quad$ *NN,VB* $\quad$ *VB,IN* $\quad$ *DT* $\quad$ *NN*

Time $\quad$ *flies* $\quad$ *like* $\quad$ *an* $\quad$ *arrow*

0 $\quad$ *1* $\quad$ *2* $\quad$ *3* $\quad$ *4* $\quad$ *5*

Complexity analysis

- What is the running time of CKY
  - as a function of input sentence length?
  - as a function of the number of rules in the grammar?

Today we will cover

how can a computer find a sentence's structure?

For context-free grammars, the (weighted) CKY algorithm can be used to find the most probable (maximum a posterior) tree given a certain grammar

Today we will cover

Linguistics	*Computer Science*
what is syntax?	how can a computer find a sentence's structure?
what is a grammar and where do they come from?	**what can parse trees be used for?**

Uses of trees

- Semantic role labeling (Weds)
- Machine translation
- **Today: measuring syntactic diversity**

Syntactic diversity

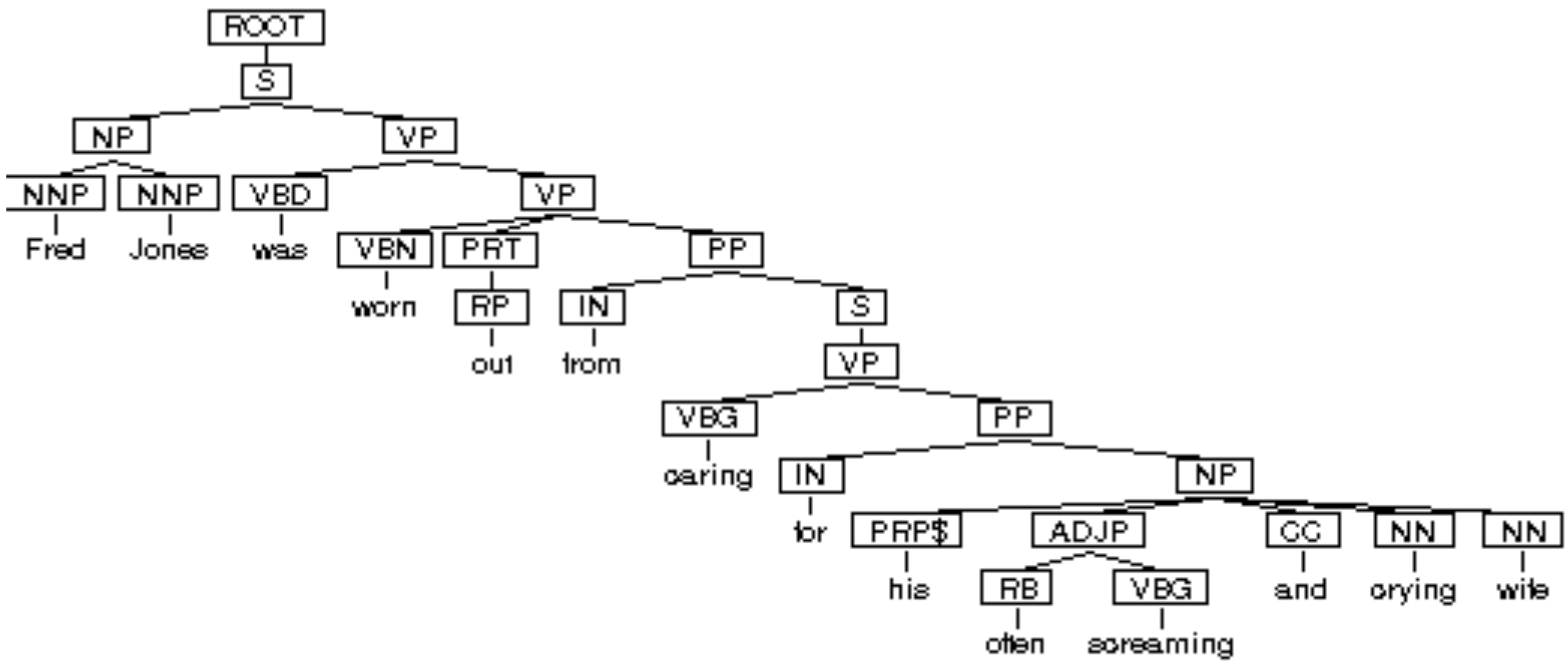
- How many ways are there to rephrase a sentence while retaining its meaning?

Fred Jones was worn out from caring for his often screaming and crying wife

- Suppose we had a paraphrase system that could rewrite this system
  - *Fred Jones was **tired** from caring for his often screaming and crying wife*
  - *Fred Jones was worn out from caring for his **frequently** screaming and crying wife*
  - *Fred Jones was worn out from caring for his often screaming and crying **spouse***
- To help train this system, we'd like a diversity metric
  - *Fred Jones' wife's frequent yelling and crying brought him to the brink of exhaustion.*

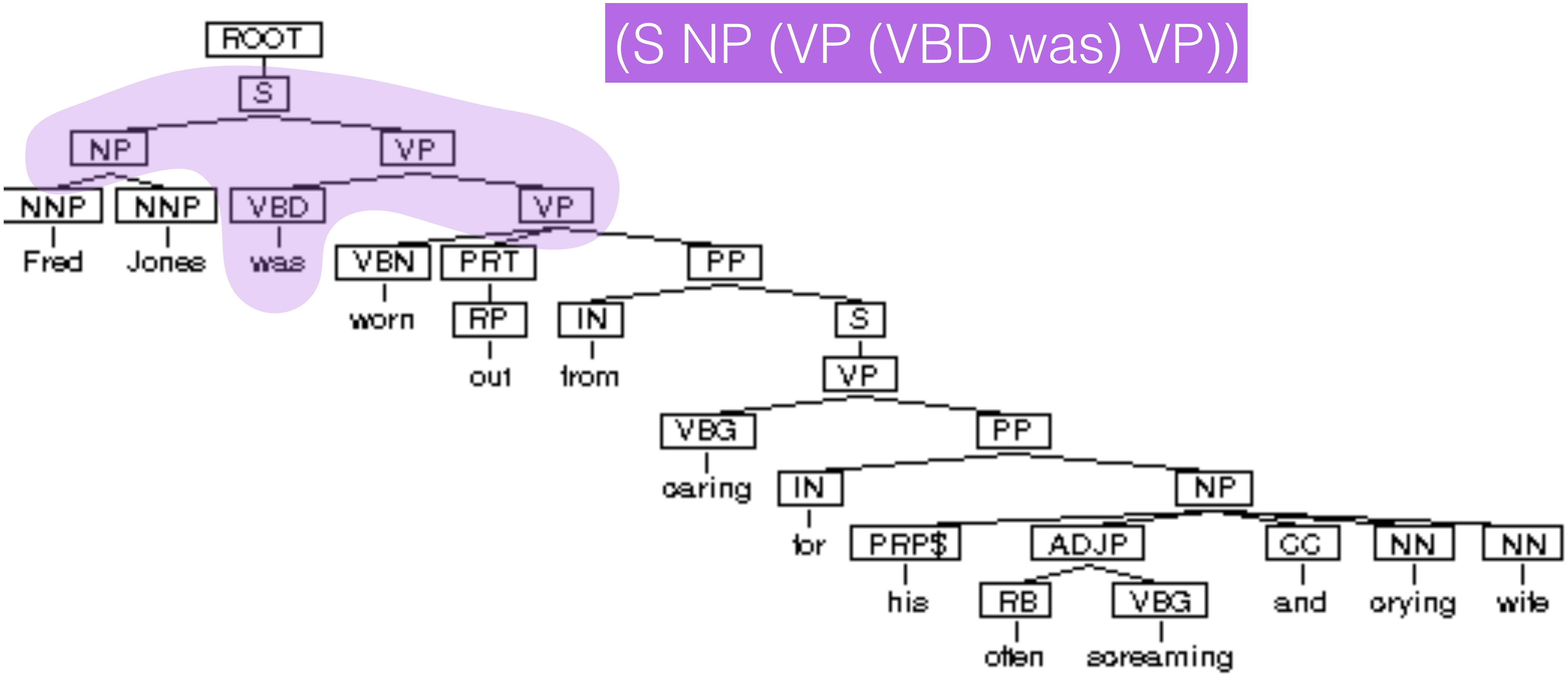
Tree kernels

- A way to compare how similar trees are by counting how many *tree fragments they have in common*



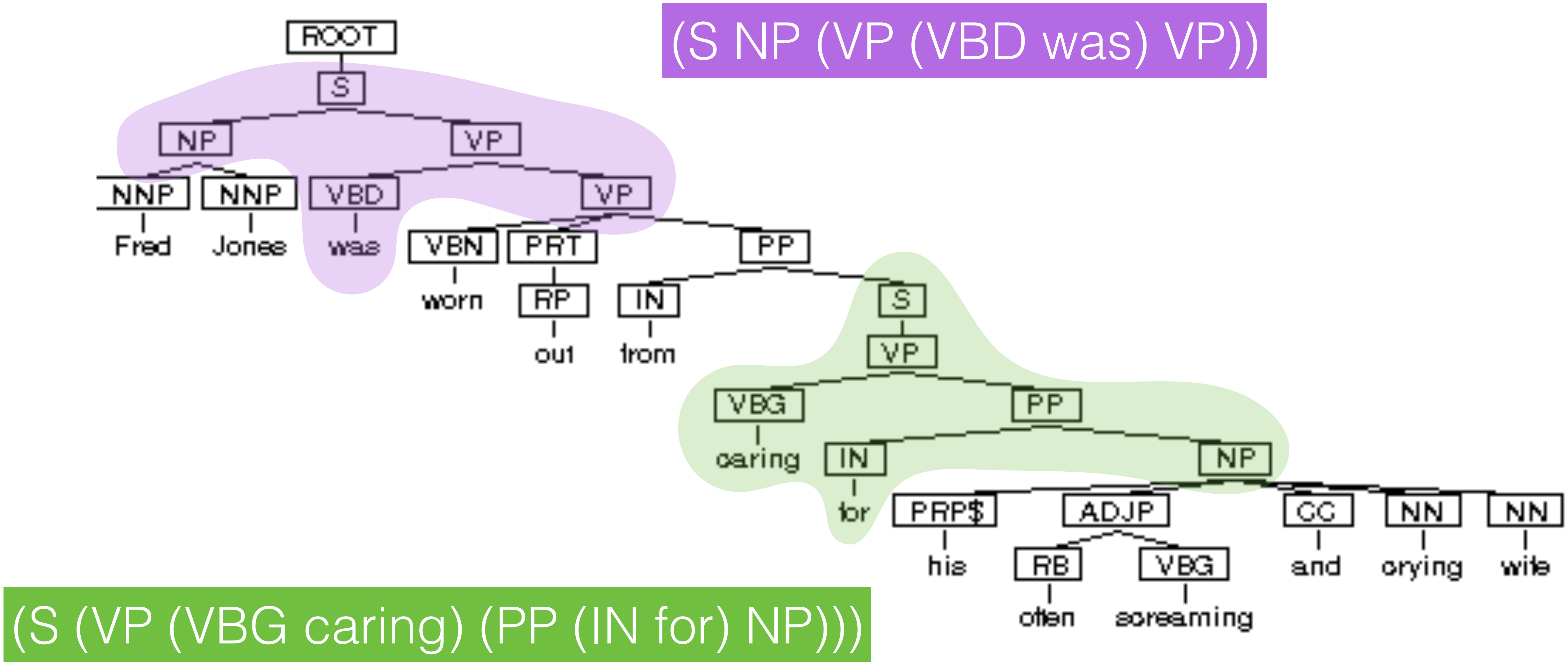
Tree kernels

- A way to compare how similar trees are by counting how many *tree fragments they have in common*



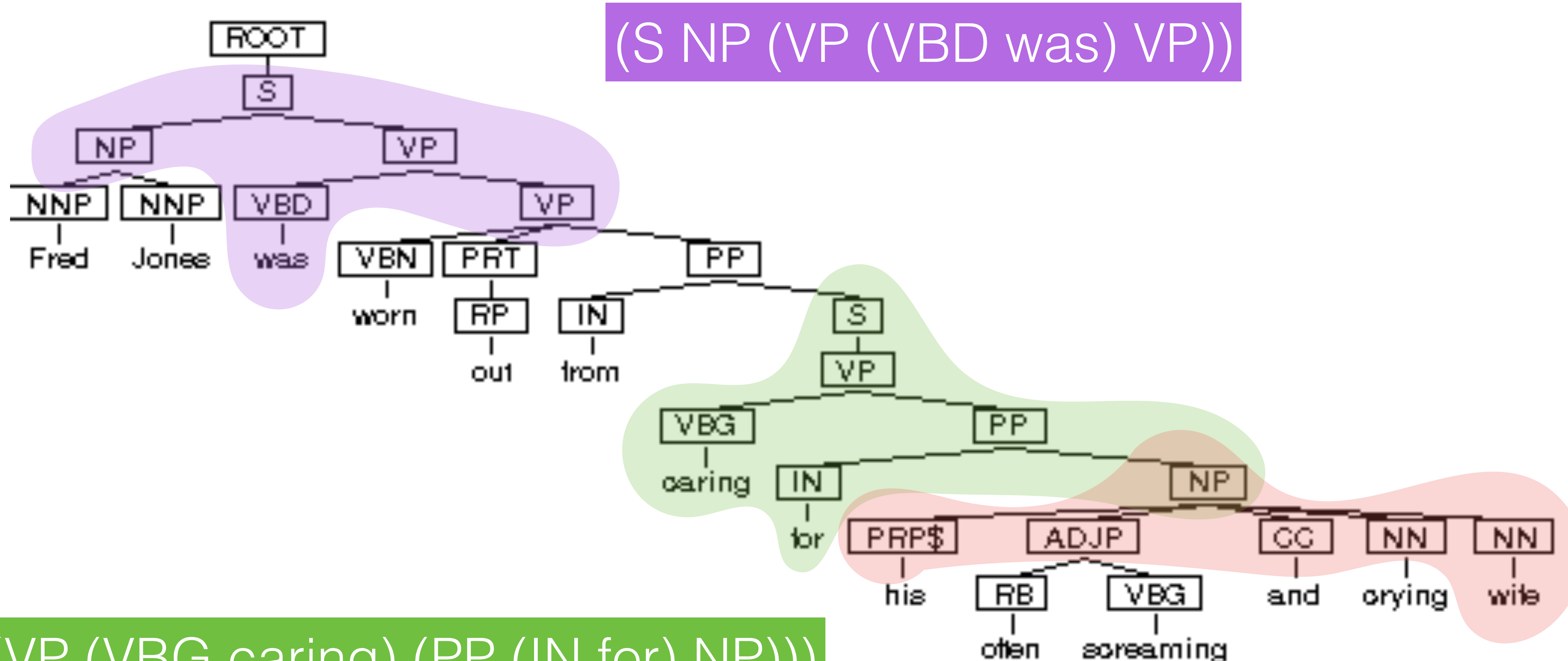
Tree kernels

- A way to compare how similar trees are by counting how many *tree fragments they have in common*



Tree kernels

- A way to compare how similar trees are by counting how many *tree fragments they have in common*

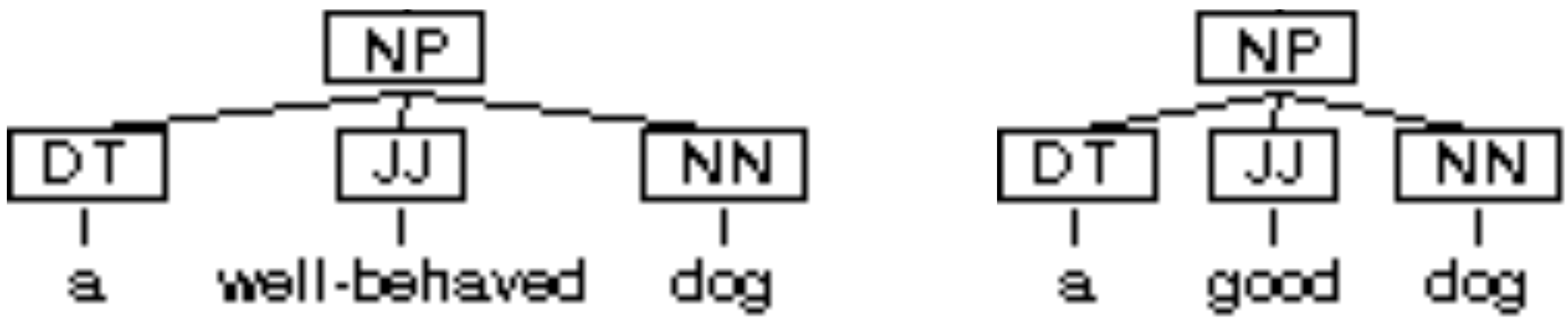


(S NP (VP (VBD was) VP))

(S (VP (VBG caring) (PP (IN for) NP)))

(NP PRP$ ADJP CC NN (NN wife))

- how many fragments?
- how many fragments in common?



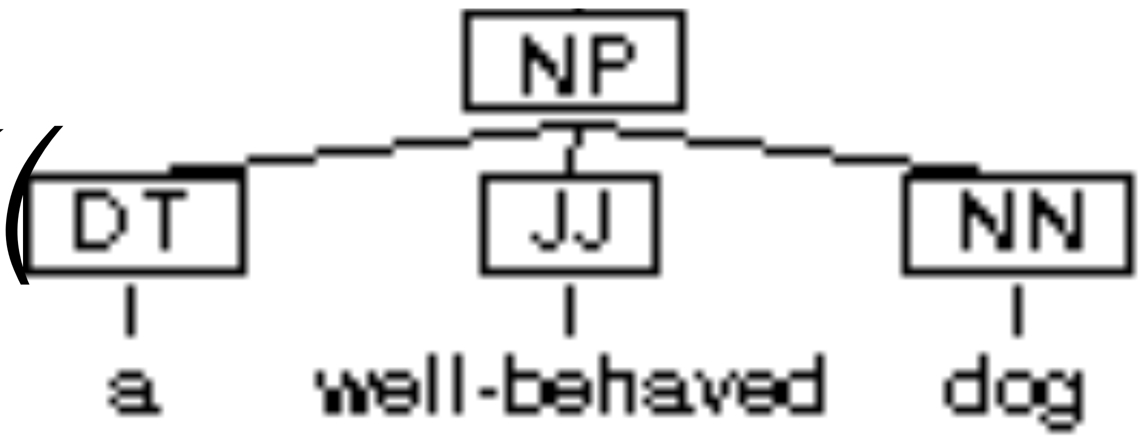
Algorithm

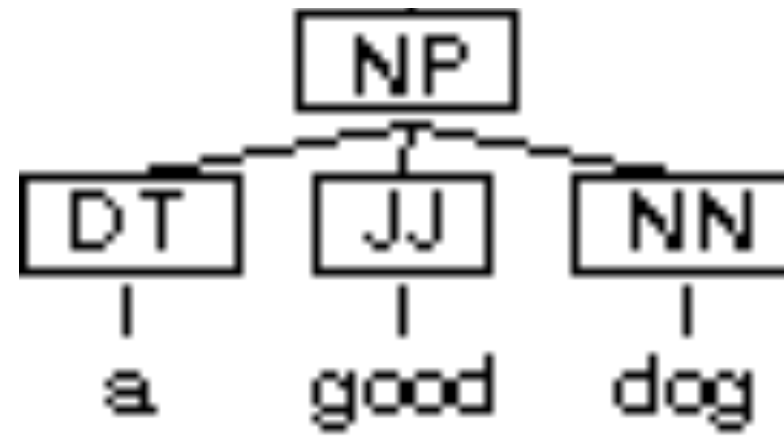
- Node score: $\Delta(n_1, n_2) =$
  - 0 if $\text{rule}(n_1) \neq \text{rule}(n_2)$
  - 1 if the rules are the same and are terminal rules
  - $\prod_{j=1}^{|n_1|} (1 + \Delta(n_{1j}, n_{2j}))$ otherwise
- Kernel score

$$K(T_1, T_2) = \sum_{n_1 \in T_1} \sum_{n_2 \in T_2} \Delta(n_1, n_2)$$

- how many fragments in common?

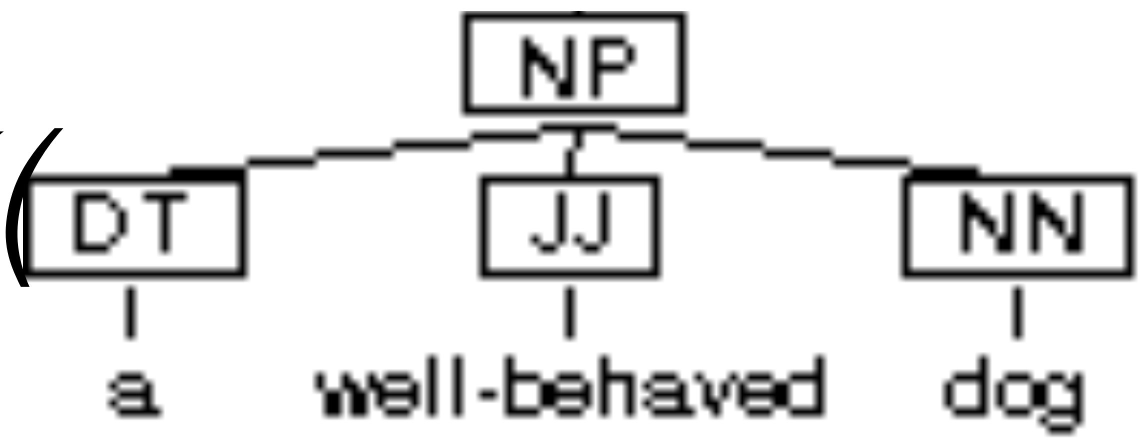
- how many fragments in common?

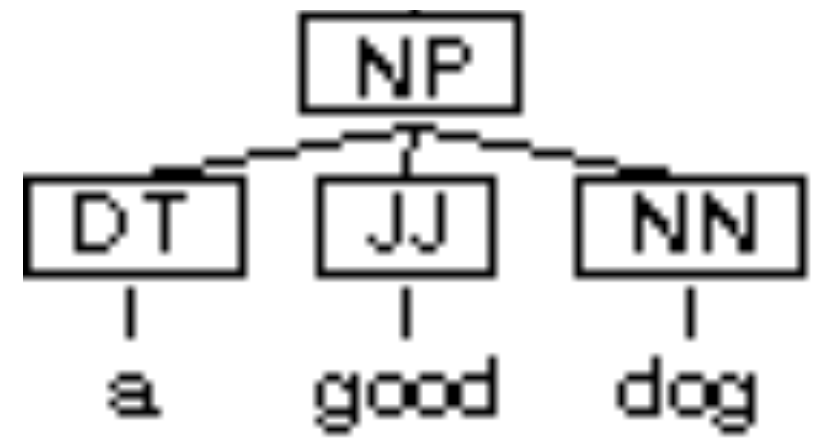
$K($ 

$,$ 

$)$

- how many fragments in common?

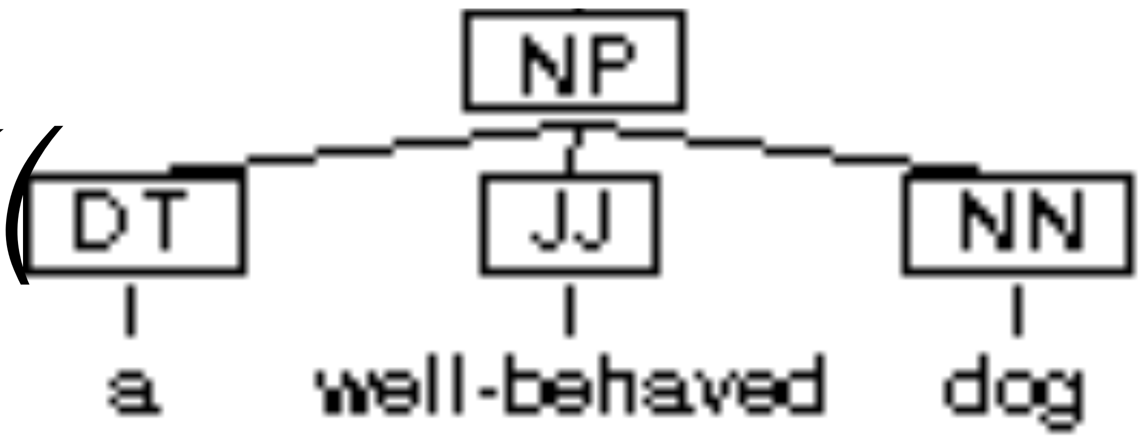
$K($ 

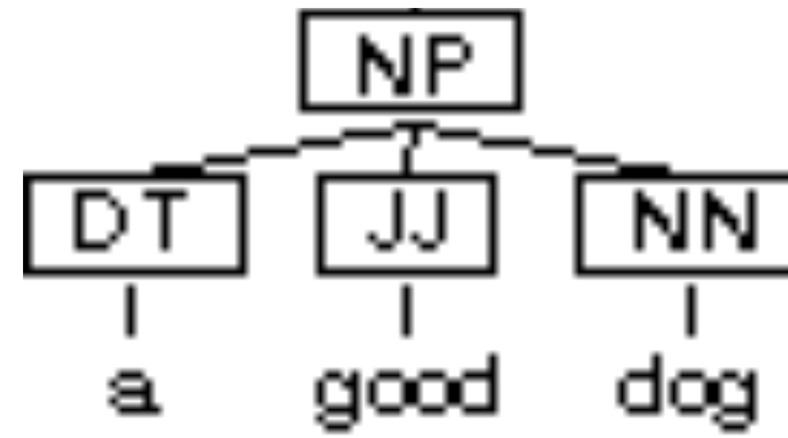
, 

$)$

$$
\begin{aligned}
&= \Delta(\mathrm{NP}_1, \mathrm{NP}_2) + \Delta(\mathrm{DT}_1, \mathrm{DT}_2) + \Delta(\mathrm{JJ}_1, \mathrm{JJ}_2) + \Delta(\mathrm{NN}_1, \mathrm{NN}_2) \\
&\quad + \Delta(\mathrm{NP}_1, \mathrm{DT}_2) + \Delta(\mathrm{NP}_1, \mathrm{JJ}_2) + \Delta(\mathrm{NP}_2, \mathrm{NN}_2) \\
&\quad + \ldots
\end{aligned}
$$

- how many fragments in common?

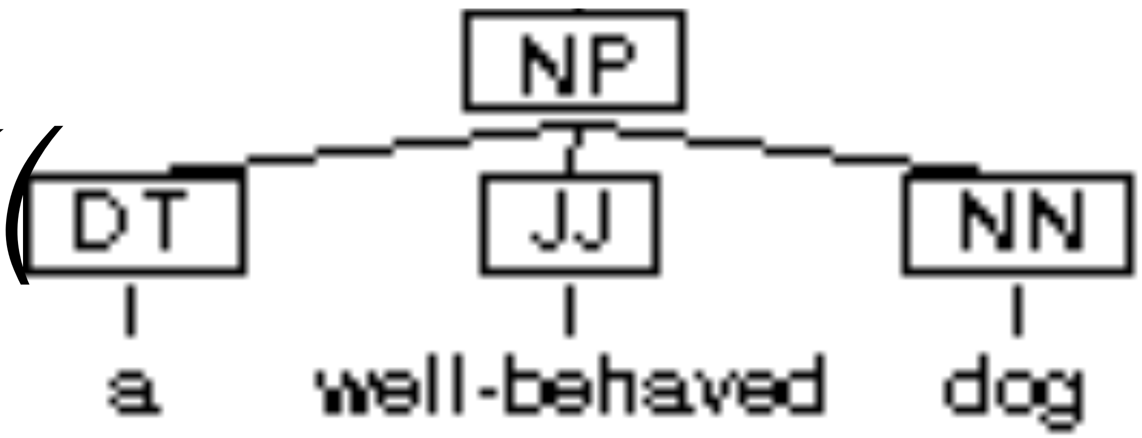
$K($ 

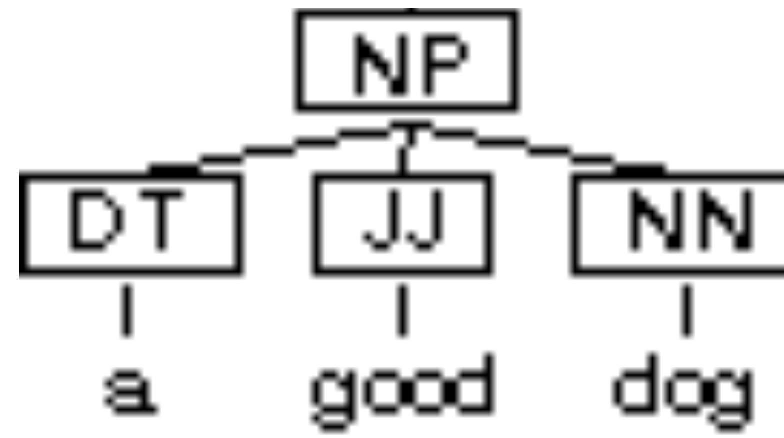
$,$ 

$)$

$$\begin{aligned}
&= \Delta(\mathrm{NP}_1, \mathrm{NP}_2) + \Delta(\mathrm{DT}_1, \mathrm{DT}_2) + \Delta(\mathrm{JJ}_1, \mathrm{JJ}_2) + \Delta(\mathrm{NN}_1, \mathrm{NN}_2) \\
&\quad + \Delta(\mathrm{NP}_1, \mathrm{DT}_2) + \Delta(\mathrm{NP}_1, \mathrm{JJ}_2) + \Delta(\mathrm{NP}_2, \mathrm{NN}_2) \\
&\quad + \ldots \\
&= \Delta(\mathrm{NP}_1, \mathrm{NP}_2) + 1 + 0 + 1
\end{aligned}$$

- how many fragments in common?

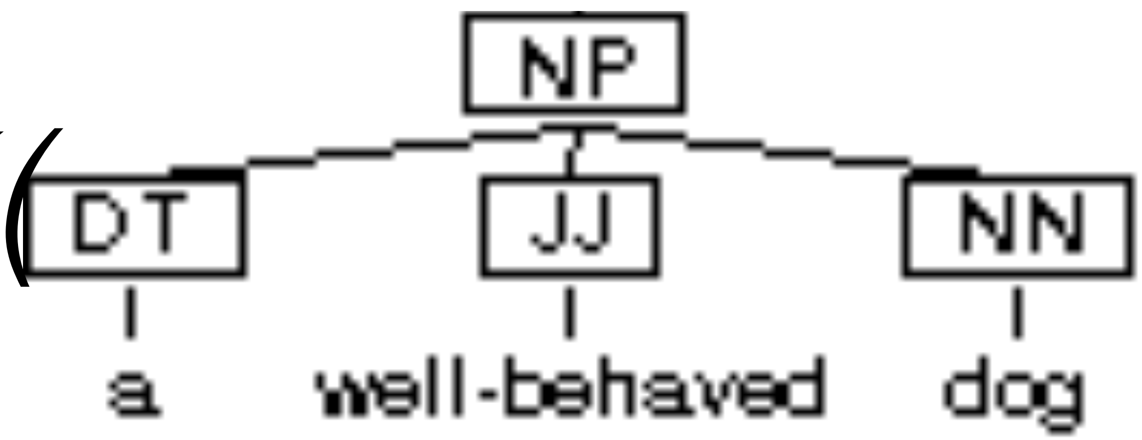
$K($ 

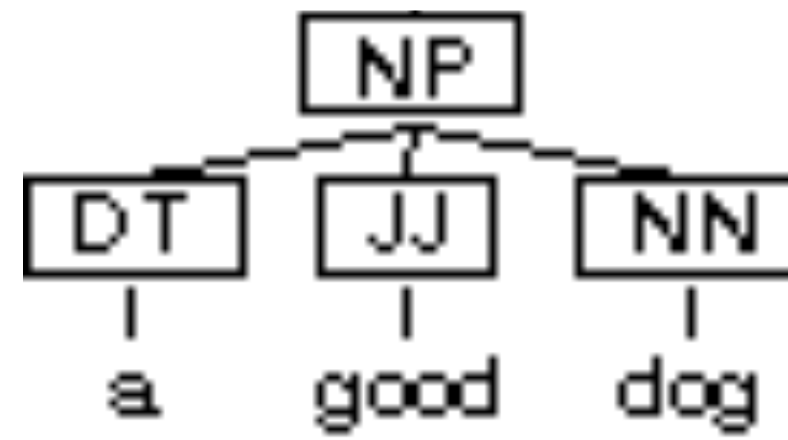
$,$ 

$)$

$$
\begin{aligned}
&= \Delta(\mathrm{NP}_1, \mathrm{NP}_2) + \Delta(\mathrm{DT}_1, \mathrm{DT}_2) + \Delta(\mathrm{JJ}_1, \mathrm{JJ}_2) + \Delta(\mathrm{NN}_1, \mathrm{NN}_2) \\
&\quad + \Delta(\mathrm{NP}_1, \mathrm{DT}_2) + \Delta(\mathrm{NP}_1, \mathrm{JJ}_2) + \Delta(\mathrm{NP}_2, \mathrm{NN}_2) \\
&\quad + \ldots \\
&= \Delta(\mathrm{NP}_1, \mathrm{NP}_2) + 1 + 0 + 1 \\
&= (1 + \Delta(\mathrm{DT}_1, \mathrm{DT}_2) \cdot (1 + \Delta(\mathrm{JJ}_1, \mathrm{JJ}_2) \cdot (1 + \Delta(\mathrm{NN}_1, \mathrm{NN}_2)) + 2
\end{aligned}
$$

- how many fragments in common?

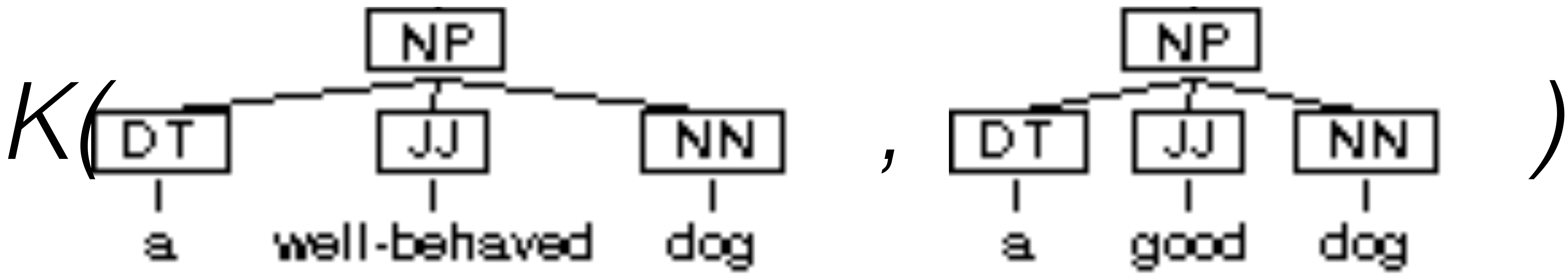
$K($ 

$,$ 

$)$

$$
\begin{aligned}
&= \Delta(\mathrm{NP}_1, \mathrm{NP}_2) + \Delta(\mathrm{DT}_1, \mathrm{DT}_2) + \Delta(\mathrm{JJ}_1, \mathrm{JJ}_2) + \Delta(\mathrm{NN}_1, \mathrm{NN}_2) \\
&\quad + \Delta(\mathrm{NP}_1, \mathrm{DT}_2) + \Delta(\mathrm{NP}_1, \mathrm{JJ}_2) + \Delta(\mathrm{NP}_2, \mathrm{NN}_2) \\
&\quad + \ldots \\
&= \Delta(\mathrm{NP}_1, \mathrm{NP}_2) + 1 + 0 + 1 \\
&= (1 + \Delta(\mathrm{DT}_1, \mathrm{DT}_2) \cdot (1 + \Delta(\mathrm{JJ}_1, \mathrm{JJ}_2) \cdot (1 + \Delta(\mathrm{NN}_1, \mathrm{NN}_2)) + 2 \\
&= (1 + 1) \cdot (1 + 0) \cdot (1 + 1) + 2
\end{aligned}
$$

- how many fragments in common?



$$
\begin{aligned}
&= \Delta(\mathrm{NP}_1, \mathrm{NP}_2) + \Delta(\mathrm{DT}_1, \mathrm{DT}_2) + \Delta(\mathrm{JJ}_1, \mathrm{JJ}_2) + \Delta(\mathrm{NN}_1, \mathrm{NN}_2) \\
&\quad + \Delta(\mathrm{NP}_1, \mathrm{DT}_2) + \Delta(\mathrm{NP}_1, \mathrm{JJ}_2) + \Delta(\mathrm{NP}_2, \mathrm{NN}_2) \\
&\quad + \ldots \\
&= \Delta(\mathrm{NP}_1, \mathrm{NP}_2) + 1 + 0 + 1 \\
&= (1 + \Delta(\mathrm{DT}_1, \mathrm{DT}_2) \cdot (1 + \Delta(\mathrm{JJ}_1, \mathrm{JJ}_2) \cdot (1 + \Delta(\mathrm{NN}_1, \mathrm{NN}_2)) + 2 \\
&= (1 + 1) \cdot (1 + 0) \cdot (1 + 1) + 2 \\
&= 6
\end{aligned}
$$

Details

Making Tree Kernels practical for Natural Language Learning

Alessandro Moschitti
Department of Computer Science
University of Rome "Tor Vergata"
Rome, Italy
moschitti@info.uniroma2.it

Abstract

In recent years tree kernels have been proposed for the automatic learning of natural language applications. Unfortunately, they show (a) an inherent super linear complexity and (b) a lower accuracy than traditional attribute/value methods.

In this paper, we show that tree kernels are very helpful in the processing of natural language as (a) we provide a simple algorithm to compute tree kernels in linear average running time and (b) our study on the classification properties of diverse tree kernels show that kernel combinations always improve the traditional methods. Experiments with Support Vector Machines on the predicate argument classification task provide empirical support to our thesis.

1 Introduction

In recent years tree kernels have been shown to be interesting approaches for the modeling of syntactic information in natural language tasks, e.g. syntactic parsing (Collins and Duffy, 2002), relation extraction (Zelenko et al., 2003), Named Entity recognition (Cumby and Roth, 2003; Culotta and Sorensen, 2004) and Semantic Parsing (Moschitti, 2004).

The main tree kernel advantage is the possibility to generate a high number of syntactic features and let the learning algorithm to select those most relevant for a specific application. In contrast, their major drawback are (a) the computational time complexity which is superlinear in the number of tree nodes and (b) the accuracy that they produce is often lower than the one provided by linear models on manually designed features.

To solve problem (a), a linear complexity algorithm for the *subtree* (ST) kernel computation, was designed in (Vishwanathan and Smola, 2002). Unfortunately, the ST set is rather poorer than the one generated by the subset tree (SST) kernel designed in (Collins and Duffy, 2002). Intuitively, an ST rooted in a node n of the target tree always contains all n's descendants until the leaves. This does not hold for the SSTs whose leaves can be internal nodes.

To solve the problem (b), a study on different tree substructure spaces should be carried out to derive the tree kernel that provide the highest accuracy. On the one hand, SSTs provide learning algorithms with richer information which may be critical to capture syntactic properties of parse trees as shown, for example, in (Zelenko et al., 2003; Moschitti, 2004). On the other hand, if the SST space contains too many irrelevant features, overfitting may occur and decrease the classification accuracy (Cumby and Roth, 2003). As a consequence, the fewer features of the ST approach may be more appropriate.

In this paper, we aim to solve the above problems. We present (a) an algorithm for the evaluation of the ST and SST kernels which runs in linear average time and (b) a study of the impact of diverse tree kernels on the accuracy of Support Vector Machines (SVMs).

Our fast algorithm computes the kernels between two syntactic parse trees in $O(m + n)$ average time, where m and n are the number of nodes in the two trees. This low complexity allows SVMs to carry out experiments on hundreds of thousands of training instances since it is not higher than the complexity of the polynomial ker-

Making Tree Kernels Practical for Natural Language Learning
Alessandro Moschitti
EACL 2006
https://www.aclweb.org/anthology/E06-1015/

Today we will cover

Parse trees are useful in a wide range of tasks

One application, tree kernels, can be used to compare how similar two trees are by looking at all possible fragments between them

what can parse trees be used for?

Today you were to learn

Linguistics	*Computer Science*
what is syntax?	how can a computer find a sentence's structure?
what is a grammar and where do they come from?	what can parse trees be used for?

Today you were to learn

Linguistics	*Computer Science*
syntax is the study of the structure of language	how can a computer find a sentence's structure?
what is a grammar and where do they come from?	what can parse trees be used for?

Today you were to learn

Linguistics	*Computer Science*
syntax is the study of the structure of language	how can a computer find a sentence's structure?
grammars usually provide generative stories and can be learned from Treebanks	what can parse trees be used for?

Summary

Linguistics

Computer Science

syntax is the study of the structure of language

trees can be produced by parsing a sentence with a grammar

grammars usually provide generative stories and can be learned from Treebanks

what can parse trees be used for?

Summary

Linguistics	*Computer Science*
syntax is the study of the structure of language	trees can be produced by parsing a sentence with a grammar
grammars usually provide generative stories and can be learned from Treebanks	**trees are useful in many applications, including testing syntactic diversity**